НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ННК "Інститут прикладного системного аналізу"
(повна назва інституту/факультету)

Системного проектування
(повна назва кафедри)

«На правах рукопису»
УДК 004:004.457

«До захисту допущено»

Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)

"___" _____ 2018 р.

Магістерська дисертація

зі спеціальності (спеціалізації) 122 – комп'ютерні науки та інформаційні технології
(код і назва спеціальності)

на тему: Мікросервіс парсингу і аналізу текстів, що отримуються з електронної медичної картки

Виконав (-ла): студент (-ка) 6 курсу, групи ДА-61м
(шифр групи)

Сутула Олександр Віталійович _____
(прізвище, ім'я, по батькові) (підпис)

Науковий керівник к. т. н., доцент Кисельов Г.Д. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант Розробка стартап-проекту к. т. н., доцент Кисельов Г.Д. _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2018 року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Інститут/факультет ННК "Інститут прикладного системного аналізу"
(повна назва)

Кафедра Системного проектування
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною (освітньо-науковою) програмою

Спеціальність (спеціалізація) 122 – комп'ютерні науки та інформаційні технології (Інформаційні технології і системи проектування)
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ А.І.Петренко
(підпис) (ініціали, прізвище)
«___»_____2018 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Сутулі Олександру Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема дисертації Мікросервіс парсингу і аналізу текстів, що отримуються з електронної медичної картки _____
_____
науковий керівник дисертації _Кисельов Генадій Дмітрович, к.т.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Строк подання студентом дисертації _____

3. Об'єкт дослідження _парсери для роботи щодо аналізу текстів, що отримуються з електронної медичної картки _____
_____

4. Предмет дослідження (Вихідні дані – для магістерської дисертації за освітньо-професійною програмою) _основні стандарти медицинської карти, їхня структура, інструменти для побудови синтаксичного аналізатора та DSL
_____

5. Перелік завдань, які потрібно розробити _____

1. Дослідження існуючих стандартів медичної картки.
2. Дослідження уже існуючих рішень щодо обраного станадрта.
3. Вибір мови та розробка парсера.
4. Розробка стартап-проекту.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу _____ __ презентація на тему «Мікросервіс парсингу і аналізу текстів, які отримуються з електронної медичної картки»_____

7. Консультанти розділів дисертації·

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розробка стартап-проекту	Кисельов Г.Д., доц.		

9. Дата видачі завдання ___01.02.2018_____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	01.02.2018	
2	Огляд стану предметної області	15.02.2018	
3	Дослідження існуючих стандартів медичної картки	05.03.2018	
4	Аналіз існуючих рішень	30.03.2018	
5	Розробка архітектури моделі парсера	10.04.2018	
6	Реалізація запропонованого методу	17.04.2018	
7	Порівняння рішення з уже існуючими парсерами	30.04.2018	
8	Отримання допуску до захисту та подача роботи в ДЕК	10.05.2018	

Студент _____ О.В. Сутула
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Г.Д. Кисельов
(підпис) (ініціали, прізвище)

· Консультантом не може бути зазначено наукового керівника

РЕФЕРАТ МАГІСТЕРСЬКОЇ ДИСЕРТАЦІЇ

виконаної на тему: Мікросервіс парсингу і аналізу текстів, які отримуються з електронної медичної картки

студентом: Сутулою Олександром Віталійовичем

Загальний обсяг роботи: 73 сторінки, 12 ілюстрацій, 19 таблиць, перелік посилань із 20 найменувань.

Актуальність теми

Галузь аналізу великих даних, зокрема медичних, стрімко розвивається, і є великий попит на інструменти що дозволяють вилучати точкову інформацію з різного роду форматів даних. Медична інформація характеризується великим об’ємом різнорідних даних та, як і будь яка система обміну даними, певною мірою доповнюється мета інформацію, через збитковості даних часто виникають ситуації коли аналіз уповільнюється в рази, а інколи взагалі стає неможливим. Ці інструменти мають бути швидкими та гнучкими для забезпечення аналізу великих об’ємів даних.

На разі, існує доволі мало рішень для парсингу повідомлень HL7 на платформі JVM, найпоширенішій платформі для розробки, а існуючі мають дуже низку швидкість парсингу точкової інформації.

Мета та задачі дослідження

Метою даної роботи є удосконалення існуючих рішень для парсинга повідомлень медичного стандарту HL7. Задачею дослідження є реалізація парсера повідомлень стандарту HL7, що є більш ефективною для обробки великих масивів даних.

Вирішення поставлених завдань та досягнуті результати

Було запропоновано розробку парсера на базі Scala та бібліотеки для побудови синтаксичного аналізатора Parboiled2, що мають достатньо можливостей для покращення роботи уже існуючого інструмента HAPI HL7 Terser для JVM платформи та Akka http для забезпення REST API та можливості впровадження як міні сервісу.

Створений парсер було порівняно з HAPI HL7 Terser на наборі даних, що моделює отримання різної інформації з повідомлення.

Об’єкт дослідження

Парсери повідомлень медичного стандарту HL7

Предмет дослідження

Граматика та структура медичного стандарту HL7

Методи дослідження

Досліджується структура повідомлення медичного стандарту HL7, його формат, особливості. Аналізуються інструменти, необхідні для створення парсера для таких повідомлень, а саме інструментарій створення синтаксичного аналізатора на основі граматик, інструменти для побудови DSL для запитів інформації.

Наукова новизна

Рішення що використовує Scala та Parboiled2 для покращення виконання задач парсингу інформації з медичного формату HL7. На разі, існує єдиний аналог для JVM, що дозволяє отримати будь-яку інформацію з повідомлення, це – HAPI HL7 Terser. За допомогою рішень, описаних в цій роботі, швидкість парсингу повідомлень збільшилась в середньому у 8 разів.

Практичне значення одержаних результатів

Розроблене рішення дозволяє отримати різнородну інформацію з повідомлення медичного стандрата HL7 на порядок швидше за існуючі аналоги на платформі JVM.

Ключові слова

Парсинг, синтаксичний аналізатор, HL7, DSL, Scala, Parboiled2, граматика.

РЕФЕРАТ МАГИСТЕРСКОЙ ДИССЕРТАЦИИ

выполненной на тему: Микросервис парсинга и анализа текстов, що получаються с електронной медицинской карточки

студентом: Сутулой Александром Виталиевичем

Общий объем работы: 73 страницы, 12 иллюстраций, 19 таблиц, перечень ссылок из 20 наименований.

Актуальность темы

Отрасль анализа больших данных, в частности медицинских, стремительно развивается, и есть большой спрос на инструменты позволяющие изымать точечную информацию из различного рода форматов данных. Медицинская информация характеризуется большим объемом разнородных данных и, как и любая система обмена данными, в определенной степени дополняется цель информацию, из-за убыточности данных часто возникают ситуации, когда анализ замедляется в разы, а иногда вообще становится невозможным. Эти инструменты должны быть быстрыми и гибкими для обеспечения анализа больших объемов данных.

На данный момент, существует довольно мало решений для парсинга сообщений HL7 на платформе JVM, распространенной платформе для разработки, а иснуючи имеют очень ряд скорость парсинга точечной информации.

Цель и задачи исследования

Целью данной работы является совершенствование существующих решений для парсинга сообщений медицинского стандарта HL7. Задачей исследования является реализация парсера сообщений стандарта HL7, что является более эффективной для обработки больших массивов данных.

Решение поставленных задач и достигнутые результаты

Было предложено разработку парсера на базе Scala и библиотеки для построения синтаксического анализатора Parboiled2, что имеют достаточно возможностей для улучшения работы уже существующего инструмента HAPI HL7 Terser для JVM платформы и Akka http для забезпення REST API и возможности внедрения мини сервиса.

Созданный парсер было сравнимо с HAPI HL7 Terser на наборе данных, моделирующий получения различной информации из сообщения.

Объект исследования

Парсеры сообщений медицинского стандарта HL7.

Предмет исследования

Грамматика и структура медицинского стандарта HL7.

Методы исследования

Исследуется структура сообщение медицинского стандарта HL7, его формат, особенности. Анализируются инструменты, необходимые для создания парсера для таких сообщений, а именно инструментарий создания синтаксического анализатора на основе грамматик, инструменты для построения DSL для запросов информации.

Научная новизна

Решение использующий Scala и Parboiled2 для улучшения выполнения задач парсинга информации из медицинского формата HL7. На данный момент, существует единственный аналог для JVM, что позволяет получить любую информацию из сообщения, это - HAPI HL7 Terser. С помощью решений, описанных в этой работе, скорость парсинга сообщений увеличилась в среднем в 8 раз.

Практическое значение полученных результатов

Разработанное решение позволяет получить разнородных информацию из сообщения медицинского стандрат HL7 на порядок быстрее существующих аналогов.

Ключевые слова

Парсинг, синтаксический анализатор, HL7, DSL, Scala, Parboiled2, грамматика.

ABSTRACT FOR MASTER'S THESIS

on: Microservice for parsing and analysis of texts received from the electronic medical card

by: Sutula Oleksandr VItalIyovich

The thesis contains 73 pages, 12 figures, 19 tables, 20 references.

Relevance

The branch of analysis of large data, in particular medical, is rapidly developing, and there is a great demand for tools that allow to extract point information from various types of data formats. Medical information is characterized by a large amount of heterogeneous data and, like any data exchange system, the information is supplemented to a certain extent with information, due to the loss of data, situations often arise where the analysis slows down at times, and sometimes even becomes impossible. These tools should be fast and flexible to provide analysis of large amounts of data.

At the moment, there are quite a few solutions for parsing HL7 messages on the JVM platform, a common development platform, and there are a lot of speed of parsing of point information.

Purpose

The purpose of this work is to improve existing solutions for parsing messages of medical standard HL7. The task of the study is the implementation of the message parser standard HL7, which is more efficient for processing large data sets.

Results

It was suggested to develop a parser based on Scala and a library for building Parboiled2 parser that have enough opportunities to improve the work of the existing HAPI HL7 Terser tool for the JVM platform and Akka http for the sake of the REST API and the possibility of implementing a mini service.

The created parser was comparable to the HAPI HL7 Terser on a data set, simulating the receipt of various information from the message.

Object of research

Message parsers of medical standard HL7.

Subject of research

Grammar and structure of medical standard HL7.

Research methods

The structure of the report of the medical standard HL7, its format, features is investigated. The tools needed to create a parser for such messages are analyzed, namely the tools for creating a grammar-based parser, tools for building DSL for information requests.

Scientific novelty

The solution uses Scala and Parboiled2 to improve the performance of information parsing tasks from medical format HL7. At the moment, there is only one analog for the JVM, which allows you to get any information from the message, it's HAPI HL7 Terser. With the help of the solutions described in this work, the message parsing speed has increased by an average of 8 times.

Practical value

The developed solution allows you to obtain heterogeneous information from the medical standard HL7 message an order of magnitude faster than existing analogues.

Keywords

Parsing, parser, HL7, DSL, Scala, Parboiled2, grammar.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

API – application programming interface, прикладний програмний інтерфейс

HL7 – Health Level 7, Сьомий Рівень медичного документообігу

JVM – java virtual machine, віртуальна машина Java

Парсер – синтаксичний аналізатор

PEG – parsing expression grammar, граматика що розбиває вираз

DSL – Domain Specific Language, мова що специфічна для домена

ВСТУП

З кожним днем проблема аналізу й обробки великих масивів даних стає більш актуальною. Можливості дослідити та використати великі дані дає змогу створити медичні експертні системи підтримки прийняття рішень які ще вчора не можна було уявити.

Медична інформація характеризується великим об’ємом різнорідних даних та, як і будь яка система обміну даними, певною мірою доповнюється мета інформацію, через збитковості даних часто виникають ситуації коли їх аналіз уповільнюється в рази, а інколи взагалі стає неможливим. Саме тому важливо мати інструменти для отримання точечної інформації з повідомлень за якомога менший час для всіх форматів даних.

В медичній сфері, надзвичайно важливий точковий аналіз результатів досліджень. Адже діагноз ставиться на основі великої кількості показників, а негативний результат одного з них може з впевненістю відкинути або підтвердити хвороби. Також, збір інформації дозволяє оцінити якість лікування, виявити неочевидні ризики для хворого, передбачити всі наслідки можливих варіантів лікування.

У березні 2003 ініціатива «Об'єднана інформатика в галузі охорони здоров'я» (CHI) оголосила про свою вимогу використання стандарту Health Level Seven (HL7) всіма федеральними агентствами охорони здоров'я як основним при обміні інформацією щодо лікування пацієнта. HL7 є основним стандартом обміну даними для клінічних повідомлень і в даний час застосовується в 90 відсотках великих лікарень [1].

На сьогоднішній час, є багато інструментів які дозволяють працювати зі стандартом HL7, але всі вони, через архітектурні рішення, призначення чи бібліотеки

на основі яких вони працюють, не можуть похвалитись високою швидкістю обробки повідомлень.

Під час створення сервісу для аналізу інформації щодо лікування інфекційних захворювань незадовільна швидкість парсингу повідомлень стала вузьким місцем обробки повідомлень, що і стало причиною пошуку альтернативних рішень.

В даній роботі буде проаналізовано стандарт HL7, порівняно парсери повідомлень цього стандарту та описано процес створення власного рішення проблеми парсингу.

1 МЕДИЧНИЙ СТАНДАРТ HL7

Стандарти формату повідомлень полегшують взаємодію з використанням загальних специфікацій кодування, інформаційних моделей для визначення відносин між елементами даних, архітектур документів і клінічних шаблонів для структурування даних по мірі їх обміну. За останні 40 років в світі розроблено велику кількість різноманітних стандартів електронної медицинию. Хоча в загальному універсального стандарту немає і різні стандарти часто реалізують різні сторони такої величезної області як медицина. Найбільш поширеними є Digital Imaging and COmmunications in Medicine (DICOM), EDIFACT, Cisco Medical Data Exchange Solution, HL7. Саме останній (HL7) показав хороші результати в порівняльних дослідженнях та на даний момент є найбільш поширеним [2].

У березні 2003 року ініціатива «Об'єднана інформатика в галузі охорони здоров'я» (CHI) оголосила вимогу, щоб всі федеральні агентства охорони здоров'я брали основні стандарти формату обміну повідомленнями:

1. Health Level Seven [HL7] Version 2.x [V2.x] - обмін інформацією щодо лікування пацієнта;
2. DICOM – для медичних зображень;
3. NCPDP – для фармацептичних повідомлень;
4. IEEE – для медичних пристроїв;
5. LOINC – для лабораторних досліджень.

Варто відзначити, що HL7, завдяки своїй спеціалізованій групі з автоматичного аналізу Point-of-Care в лабораторії, також розробила стандарти обміну повідомленнями для пристроїв лабораторної автоматизації (наприклад, роботи та лабораторні прилади) і контрольних пристроїв для точок обслуговування (наприклад, монітори рівня глюкози в крові). Ці стандарти знаходяться в процесі включення в стандарти IEEE і в кінцевому підсумку стануть стандартами Міжнародної організації зі стандартизації (ISO).

Серія HL7 V2.x є основним стандартом обміну даними для клінічних повідомлень і в даний час застосовується в 90 відсотках великих лікарень Сполучених Штатів Америки та в близько 12 країнах Європейского Союзу [1].

Health Level 7 (Сьомий Рівень медичного документообігу) - стандарт обміну, управління та інтеграції електронної медичної інформації. Сьомим рівнем система названа за аналогією з сімома рівнями взаємодії відкритих систем, Open Systems Interconnection або OSI. Тобто це процеси найвищого рівня [3].

Сьомий рівень підтримує виконання таких завдань як:

1. Структурування переданих даних;
2. Можливості проектування систем;
3. Досягнення узгодженості передач;
4. Безпека;
5. Ідентифікація учасників;
6. Доступність.

1.1 Складові стандарту HL7

Основний стек технологій стандарту це:

1. Еталонна інформаційна модель (Reference Information Model - RIM);
2. Розкадрування (Storyboard);
3. Словники;
4. Визначник ієрархічної структури повідомлення (Hierarchial Message Descriptor);
5. Система Електронної історії хвороби (Electronic Health Record Systems);
6. Арден синтакс;
7. Архітектура Клінічного документа [3].

1.1.1 Інформаційна модель

Еталонна Інформаційна Модель є базовим поняттям для всього HL7, основним джерелом змісту даних усіх повідомлень і документів. Елементи інформаційної моделі це класи, переходи станів класів, типи даних і накладені обмеження - використовують систем, концепцій та графічне вираження мови UML.

1.1.2 Розкадрування

Концепція розкадрування (storyboard) дозволяє представити засобами HL7 значущі моменти передачі повідомлень як кадри. У кожному кадрі описані ключові учасники і їх взаємодія. Кожна взаємодія описується розкадровкою (в UML діаграма послідовностей). Засобами RIM і розкадровкою можливо виразити високо персоніфіковану історію хворого [3].

1.1.3 Словники

Представлені у вигляді тезаурусів або навіть онтологій опису специфіки предметних областей. Атрибут в RIM-описі може бути елементом словника.

Словниками можуть бути:

1. Таблиця побудована на принципах метатезауруса UMLS описана засобами HL7;
2. LOINC, SNOMED, HIPAA, місцеві, національні словники.

1.1.4 Hierarchial Message Descriptor

Це визначник ієрархічної структури повідомлення. Принципи HMD:

1. Система передачі повинна розуміти генезис класів;
2. Повідомлення при передачі шикуються в лінійну структуровану послідовність [3].

1.1.5 Система Електронної історії хвороби

Опис повного функціоналу системи Електронної історії хвороби складається з розділів:

1. Управління наданням медичної допомоги (Care Management);
2. Клінічний документообіг (Clinical Support);
3. Інформаційна інфраструктура (Information Infrastructure).

1.1.6 Арден синтаксис

Специфікація прийнята HL7 для визначення і поширення медичних знань. Арден синтаксис є мовою Медичних логічних модулів (МЛМ - Medical Logic Modules) кодування медичних знань. Кожен МЛМ містить достатню інформацію для прийняття медичного рішення. МЛМ використовується для генерації сигналів тривоги, розуміння медичних даних, діагностики, фільтрації медичних даних і адміністративних завдань. При певних умовах може бути розроблена комп'ютерна програма (event monitor) генеруюча експертну підтримку. МЛМ може бути пов'язаний з іншими МЛМ і утворювати мережу [3].

1.1.7 Clinical Document Architecture

Архітектура клінічного документа це стандарт сфери HL7, затверджений ISO (ISO / HL7 27932: 2009 Data Exchange Standards - HL7 Clinical Document Architecture, Release 2). Даний стандарт повністю визначає синтаксис і набір структур даних дозволяють повністю описати семантику будь-якого клінічного документа. В основу архітектури покладено мову XML [3].

При створенні клінічного документа його розмітка, структура і семантика береться з опису Архітектури клінічного документа. Сама специфікація виходить на основі довідника даних RIM. Клінічний документ по CDA є повним інформаційним об'єктом, з повністю визначеними компонентами. Додатково він може містити текст, зображення, звук і інший мультимедійний зміст [3].

Спочатку HL7 починав розвиватися як стандарт повідомлень. Клінічний документ має авторство, історію змін, він стабільний і призначений для сприйняття людиною. У свою чергу повідомлення призначене для читання комп'ютером, існує тільки під час передачі документа [3].

Сам клінічний документ описаний на мові XML, але також може містити дані інших типів, такі як аудіо / відео інформацію, бінарні дані зображень, цифрові підписи [3].

В секціях клінічного документа можна висловити клінічні вирази, такі як виконані процедури, поточний стан хворого, адміністративні розпорядження, небажані події і фактори. Клінічний документ складається з заголовка і тіла. У заголовку можна висловити складну систему авторів, виконавців, відповідальності, поточний стан документа, доступ до нього. Високорівневе уявлення всіх виразних можливостей заголовка задається схемою UML. Тіло клінічного документа містить клінічну запис / звіт (clinical report), зібраний з секцій (section) [3].

Одна з цілей CDA порівнянність клінічного документа, що дозволяє організувати роботу з ними [3].

Типи даних ув'язнених в секцію можуть бути простими, такими як наприклад цілі числа або даними складної тимчасової системи (наприклад general timing specification). У секції можна використовувати раніше певні концепти (структурні типи даних) які заповнюються з плином часу і розвитком подій [3].

1.2 Граматика стандарту HL7

1.2.1 Структура повідомлення

Повідомлення HL7 використовуються для передачі електронних даних між розрізненими системами охорони здоров'я. Кожне повідомлення HL7 відправляє інформацію про конкретну подію, таку як прийом пацієнта [4].

Повідомлення HL7 складається з одного або декількох сегментів. Кожен сегмент складається з одного або декількох композитів, також званих полями.

1.2.2 Сегменти HL7

У повідомленні HL7 кожен сегмент повідомлення містить одну конкретну категорію інформації, таку як інформація про пацієнта або дані про відвідування пацієнта. У повідомленні HL7 ім'я кожного сегмента в повідомленні визначається першим полем сегмента, яке завжди має три символи. Різні типи повідомлень HL7 містять різні сегменти HL7 [4]. Ось приклад типового повідомлення HL7:

MSH | ^ ~ \ & | EPIC | EPICADT | SMS | SMSADT | 199912271408 | CHARRIS | ADT ^ A04 | 1817457 | D | 2,5 |

PID || 0493575 ^^^ 2 ^ ID 1 | 454721 || DOE ^ JOHN ^^^^ | DOE ^ JOHN ^^^^ | 19480203 | M || B | 254 MYSTREET AVE ^^ MYTOWN ^ OH ^ 44123 ^ США || (216) 123-4567 ||| M | NON | 400003403 ~ 1129086 |

NK1 || ROE ^ MARIE ^^^^ | SPO || (216) 123-4567 || EC ||||||||||||||||||||||||

PV1 || O | 168 ~ 219 ~ C ~ PMA ^^^^^^^^ |||| 277 ^ ALLEN MYLASTNAME ^ BONNIE ^^^^ |||||||| || 2688684 |||||||||||||||||||| 199912271408 |||||| 002376853

Сегменти HL7 в цьому прикладі містять наступну інформацію:

1. Сегмент MSH (Message Header) містить інформацію про саме повідомлення. Ця інформація включає відправника і одержувача повідомлення, тип повідомлення, а також дату і час його відправлення. Кожне повідомлення HL7 визначає MSH як перший сегмент [4];

2. Сегмент PID (інформація про пацієнта) містить демографічну інформацію про пацієнта, таку як ім'я, ідентифікатор пацієнта і адреса [4];
3. Сегмент NK1 (Next of Kin) містить контактну інформацію для найближчих родичів пацієнта [4];
4. Сегмент PV1 (Відвідування пацієнта) містить інформацію про перебування пацієнта в лікарні, такому як призначене місце і референт [4].

Більше 120 різних сегментів HL7 доступні для використання в повідомленнях HL7.

1.2.3 Композит HL7

Кожен сегмент повідомлення HL7 складається з одного або декількох композитів (також званих полями). За замовчуванням, | (Pipe) використовується для відділення одного композиту від іншого [4].

Композит може бути примітивним типом даних (наприклад, символьним рядком або числом) або може містити інші композити [4].

Якщо композит містить інші композити, ці субкомпозіти (або підполя) зазвичай поділяються символами ^. Якщо суб-композит також містить композити, ці суб-композити зазвичай поділяються символами &. Суб-субкомпозіти повинні бути примітивними типами даних [4].

Приклад типового композиту:

PID || 0493575 ^^^ 2 ^ ID 1 | 454721 || DOE ^ JOHN ^^^^ | DOE ^ JOHN ^^^^ | 19480203 | M || B | 254 MYSTREET AVE ^^ MYTOWN ^ OH ^ 44123 ^ США || (216) 123-4567 ||| M | NON | 400003403 ~ 1129086 |

У цьому сегменті п'ятий композит - це ім'я пацієнта, яке є DOE ^ JOHN ^^^^. (Чотири символу ^^^^ в кінці цього композиту вказують, що він має в цілому шість суб-композитів і що визначені тільки перші два субкомпозита.) У цьому композиті DOE представляє прізвище пацієнта, а JOHN - це ім'я пацієнта [4].

Щоб бути максимально гнучкими і досягти консенсусу, комітети HL7 були змушені визначати багато сегментів полів як необов'язкові. Недоліком цього рішення є те, немає впевненості, що конкретна інформація буде присутній в даному повідомленні. Це одна з причин, по якій одне й те саме повідомлення може мати відчутні відмінності від постачальника до постачальника [4].

1.2.4 Позначення граматики сегмента HL7

За угодою стандартна дескрипція граматики сегмента визначена, щоб забезпечити зручний спосіб вказівки сегментів, які можуть бути включені в конкретний тип повідомлення HL7. Багато постачальників використовують цю граматичну нотацію сегмента, щоб вказати, які сегменти можуть бути включені в кожен тип повідомлення HL7, яке вони відправляють [4].

У стандартній граматичної ноті сегмента сегменти перераховані зліва направо, починаючи з першого сегмента в повідомленні.

MSH|^~\&|EPIC|EPICADT|SMS|SMSADT|199912271408|CHARRIS|ADT^A04|1817457|D|2.5|

PID||0493575^^^2^ID 1|454721||DOE^JOHN^^^^|DOE^JOHN^^^^|19480203|M||B|254 MYSTREET AVE^^MYTOWN^OH^44123^USA||(216)123-4567|||M|NON|400003403~1129086|

NK1||ROE^MARIE^^^^|SPO||(216)123-4567||EC|||||||||||||||||||||||

PV1||O|168 ~219~C~PMA^^^^^^^^^^||||277^ALLEN MYLASTNAME^BONNIE^^^^|||||||||| ||2688684|||||||||||||||||||||||199912271408||||||002376853

Позначення граматики сегмента для цього повідомлення:

MSH PID NK1 PV1

В позначенні граматики сегмента необов'язкові сегменти укладені в квадратні дужки []. Наприклад, якщо сегмент PV2 (відвідування пацієнта - додаткова інформація) може бути необов'язково включений в повідомлення, такі як показані вище, граматика сегмента виглядає наступним чином [4]:

MSH PID NK1 PV1 [PV2]

Повторювані сегменти укладені у фігурні дужки {}. Наприклад, якщо кілька сегментів NK1 можуть бути включені, граматика сегмента виглядає наступним чином [4]:

MSH PID {NK1} PV1 [PV2]

Сегмент, який є необов'язковим і може бути повторений, укладений у квадратні дужки і фігурні дужки. Наприклад, якщо сегмент NK1 є необов'язковим, але може бути повторений, граматика сегмента виглядає наступним чином [4]:

MSH PID [{NK1}] PV1 [PV2]

Необов'язкові або повторювані сегментні групи вказуються шляхом вказівки декількох імен сегментів всередині квадратних дужок або фігурних дужок. Наприклад, повідомлення Lab Result [4]:

MSH|^~\&|MESA_RPT_MGR|EAST_RADIOLOGY|REPOSITORY|XYZ|||ORU^R01|MESA3b78 1ae8|P|2.5|||||||

PID|||CR3^^^ADT1||CRTHREE^PAUL||||||||||||PatientAcct||||||||||||

OBR||||4550|||20010501141500.0000|||||||||||||||||F|||||||||||||||||

OBX|1|HD|SR Instance UID||1.113654.1.2001.30.2.1||||||F||||||

OBX|2|TX|SR Text||Lungs expanded and clear. Conclusions Normal PA chest x-ray.||||||F||||||

OBR||||4551|||20010501141500|||||||||||||||||F|||||||||||||||||

OBX|1|HD|SR Instance UID||1.113654.1.2001.10.2.1.603||||||F||||||

OBX|2|HD|Study Instance UID|1|1.113654.1.2001.10||||||F||||||

OBX|3|HD|Series Instance UID|1|1.113654.1.2001.10.1||||||F||||||

Це повідомлення може містити одну або кілька необов'язкових груп результатів з кожною групою результатів, що складається з одного повідомлення OBR і одного або декількох повідомлень OBX. Позначення граматики сегмента для цього повідомлення [4]:

MSH PID [{OBR {OBX}}]

Тут OBR {OBX} - це позначення граматики сегмента для групи результатів. Група результатів додається як в квадратних, так і в фігурних дужках, оскільки вона є необов'язковою і може бути повторена [4].

1.2.5 Групи сегментів HL7

Деякі типи повідомлень підтримують групи сегментів. Група сегментів являє собою набір сегментів, які завжди відображаються разом. Деякі сегментні групи можуть бути необов'язковими або повторюватися [4].

Наприклад, повідомлення типу ORU ^ R01 (Lab Result) можуть містити одну або кілька груп сегментів групи результатів. Кожна група результатів складається з одного сегмента OBR (запиту спостереження) і одного або декількох сегментів OBX (спостереження / результат) [4]. Наприклад, наступне повідомлення містить дві групи сегментів групи результатів:

MSH|^~\&|MESA_RPT_MGR|EAST_RADIOLOGY|REPOSITORY|XYZ|||ORU^R01|MESA3b78 1ae8|P|2.5|||||||

PID|||CR3^^^ADT1||CRTHREE^PAUL|||||||||||PatientAcct||||||||||

OBR||||4550|||20010501141500.0000|||||||||||||||F|||||||||||||||

OBX|1|HD|SR Instance UID||1.113654.1.2001.30.2.1||||||F||||||

OBX|2|TX|SR Text||Lungs expanded and clear. Conclusions Normal PA chest x-ray.||||||F||||||

OBR||||4551|||20010501141500||||||||||||||||F||||||||||||||||

OBX|1|HD|SR Instance UID||1.113654.1.2001.10.2.1.603||||||F||||||

OBX|2|HD|Study Instance UID|1|1.113654.1.2001.10||||||F||||||

OBX|3|HD|Series Instance UID|1|1.113654.1.2001.10.1||||||F||||||

1.2.6 Існуючі “null” поля

У сегментах важливо відзначити, що існує різниця між полями, для яких не визначено значення для них, і полями, для яких визначено значення null [4].

Якщо поле не визначене, нема нічого між двома | символами, які обмежують поле:

OBR|||||||||

Нульове значення вказується двома символами подвійних лапок:

OBR|||||””||||

Поле, для якого визначено значення null, часто згадується як існуюче зі значенням null. Існуючі, але нульові поля використовуються в багатьох додатках [4].

1.2.7 Кастомні сегменти: Z-сегменти HL7

Якщо повідомлення HL7 містить налаштовані дані, які не можуть бути включені в якийсь сегмент, визначений для його типу повідомлення, ви можете створити власний сегмент для передачі цих даних[4].

За угодою всі призначені для користувача сегменти починаються з літери Z. Наприклад, може бути створений сегмент ZPD, який буде містити налаштовану

демографічну інформацію пацієнта. Ці призначені для користувача сегменти відомі як Z-сегменти [4].

Додатки, що обробляють повідомлення HL7, зазвичай налаштовані на ігнорування Z-сегментів HL7, поведінка яких не формалізована [4].

Деякі постачальники використовують Z-сегменти HL7 для передачі інформації, яка також може зберігатися в стандартному сегменті HL7. Це може ускладнити обробку повідомлень [4].

1.2.8 Повторювані і необов'язкові сегменти

Деякі типи повідомлень HL7 дозволяють повторювати сегмент один або кілька разів або дозволяють зробити сегмент необов'язковим [4].

Наприклад, повторювані сегменти корисні для повідомлень, що містять контактну інформацію, оскільки вони дозволяють надавати більш одного контакту. Наприклад, кілька NK1 (Next of Kin) сегментів можуть бути надані, якщо пацієнт має більше однієї людини, з яким можна зв'язатися в разі надзвичайної ситуації [4].

Необов'язкові сегменти корисні для надання інформації, яка не надається у всіх повідомленнях. Наприклад, сегмент AL1 (інформація про алергії пацієнта) може бути включений, якщо у пацієнта є алергія [4].

MSH|^~\&|EPIC|EPICADT|SMS|SMSADT|199912271408|CHARRIS|ADT^A04|1817457|D|2.5|

PID||0493575^^^2^ID 1|454721||DOE^JOHN^^^^|DOE^JOHN^^^^|19480203|M||B|254 MYSTREET AVE^^MYTOWN^OH^44123^USA||(216)123-4567|||M|NON|400003403~1129086|999-|

NK1||ROE^MARIE^^^^|SPO||(216)123-4567||EC|||||||||||||||||||||||

NK1||DOE^JOHN ^^^^|SPO||(216)123-4567||EC|||||||||||||||||||||||

NK1||DOE^ROBERT ^^^^|SPO||(216)123-4568||EC|||||||||||||||||||||||

PV1||O|168 ~219~C~PMA^^^^^^^^^^||||277^ALLEN MYLASTNAME^BONNIE^^^^|||||||||| ||2688684|||||||||||||||||||||||199912271408||||||002376853

1.2.9 Визначення типу повідомлення HL7

Кожне повідомлення HL7 має конкретний тип повідомлення. Цей тип повідомлення HL7 вказує, яка інформація, пов'язана зі здоров'ям, надається в електронному листі. Тип повідомлення також визначає, які сегменти можуть бути включені як частина повідомлення [4].

Щоб визначити тип повідомлення HL7, аналізується його сегмент MSH. Тип повідомлення зазвичай є дев'ятим полем цього сегмента. Наприклад, розглянемо цей сегмент MSH, який ви бачили раніше:

MSH|^~\&|EPIC|EPICADT|SMS|SMSADT|199912271408|CHARRIS|ADT^A04|1817457|D|2.5|

Тут тип повідомлення HL7 - “ADT ^ A04”, що означає «Зареєструвати пацієнта».

1.2.10 Символи роздільники HL7

Деякі спеціальні символи відокремлюють один композит в сегменті від іншого або відокремлюють один субкомпозіт від іншого. Ці спеціальні символи відомі як символи-роздільники [4].

Таблиця 1.1 Символи роздільники за замовчуванням, що використовуються в HL7

Символ	Призначення
0x0D	Позначає кінець кожного сегмента
|	Композитний роздільник
^	Суб-композитний роздільник.

&	Суб-суб-композитний роздільник
~	Відділяє повторювані поля
\	Escape-символ

1.2.11 Перевизначення роздільників HL7

Повідомлення HL7 може вибрати використання перевизначень HL7 символів-роздільників для використання різних символів роздільника, якщо стандартні символи не підходять для використання [4].

У будь-якому повідомленні HL7 символи роздільник, що використовуються цим повідомленням, вказані як перше поле сегмента MSH. Якщо в повідомленні використовуються символи роздільник за замовчуванням, сегмент MSH повідомлення починається з наступного [4]:

MSH | ^ ~ \ &

Символи, які йдуть за MSH, вказують символи роздільник, що використовуються в цьому повідомленні. Вони в порядку:

1. Символ роздільник, який відокремлює один композит від іншого (за замовчуванням - |)[4];
2. Символ роздільник, який відокремлює один субкомпозіт від іншого (за замовчуванням ^)[4];
3. Символ роздільник, який розділяє повторювані поля (за замовчуванням ~)[4];
4. Символ роздільник, який вказує початок або кінець escape-послідовності (за замовчуванням \)[4];
5. Символ роздільник, який відокремлює один суб-суб-композит від іншого (за замовчуванням &)[4].

Якщо вхідне повідомлення HL7 використовує різні символи роздільник, ці перші п'ять символів сегмента MSH визначають використовувані розділові символи[4].

1.3 Висновки

Стандарти передачі медичної інформації набувають нового значення в наш час. Адже новітні технології дають неосяжні можливості покращення якості медицини, за рахунок збору та обробки інформації, яка неможлива без якісних та поширених стандартів. Зараз, найбільш поширеним медичним стандартом передачі даних є HL7, що дозволяє передавати будь-яку інформацію про пацієнта, яка потім може бути об’єднана в медичну карту.

HL7 має просту граматику, яку досить легко можна представити засобами сучасних інструментів для побудови граматики. Але, в той же час,простого та швидкого засобу для парсингу HL7 повідомлень немає, а ті що є, не дозволяють обробити великі масиви даних за прийнятний час.Також, для парсеру необхідна зручна та природня DSL, яка допоможе без зайвих проблем дістати будь-яку інформацію з повідомлення.

2 ВИБІР ІНСТРУМЕНТІВ ДЛЯ СТВОРЕННЯ ПАРСЕРА ПОВІДОМЛЕНЬ HL7

Для створення ефективного парсера повідомлень HL7 стандарта, необхідно: обрати мову програмування, інструменти опису граматики та створення Domain Specific Language.

Мова Scala була обрана як поширена мова, що об’єднує в собі можливості функціонального та об’єктно-орієнтованого програмування, також перевагою її є те, що вона використовує JVM і має зворотню сумісність з Java. А отже, поширені системи зможуть легко використовувати цю бібліотеку використовуючи всі переваги системи на JVM. [5]

Так як, система планується як бібліотека з можливістю використання як мікросервіс, а також ключові критерії будуть швидкість та зручність використання, то інструменти мають бути з необхідними властивостями. Необхідно проаналізувати існуючі інструменти для створення синтаксичного аналізатора, що підтримується мовою Scala та має необхідні властивості для парсингу повідомлень даного стандарта. Наступним кроком необхідно визначитись із DSL, а саме: як буде виглядати мова запитів різноманітної інформації з повідомлення, за допомогою чого вона буде реалізована.

2.1 Інструмент парсингу граматики для повідомлень HL7 на мові Scala

У цьому розділі порівняно можливості написання синтаксичного аналізатора. А саме, розглянемо такі варіанти:

- використання RegExp;
- стандартна бібліотека Scala Parser Combinators;
- Parboiled1;
- Parboiled2.

Parboiled - бібліотека, що дозволяє з легкістю розбирати (парсити) мови розмітки (такі як HTML, XML або JSON), мови програмування, конфігураційні файли, логи, текстові протоколи і взагалі що завгодно текстове. Parboiled необхідна для розробки своєї предметно-орієнтованої мови (DSL): з її допомогою ви зможете швидко отримати абстрактне синтаксичне дерево і, використваши патерн інтерпретатор, виконувати команди вашого доменної мови. [6]

На даний момент існує кілька версій даної бібліотеки:

- Parboiled for Java - найперша версія бібліотеки. Написана Маттіасом Доеніцем (Matthias Doeniz) на Java і для Java. До сих пір користується популярністю, хоч і знаходиться в стані «end of life»; [7]
- Parboiled - бібліотека, тепер уже більш відома як Parboiled1, з'явилася на світ після того, як Маттіас перейнявся Scala. Він зробив Scala-фронтенд для Parboiled, а також перестав підтримку Java-версії. З виходом Parboiled2 потихеньку перестає підтримуватися і Scala-версія Parboiled1, проте не дивлячись на це, списувати її з рахунків поки що не варто: Parboiled2 поки не має того функціоналу, що є у Parboiled1, Parboiled1 все ще використовується набагато ширше, ніж Parboiled2; [8]
- Parboiled2 - новітня версія бібліотеки, усуває ряд недоліків PB1. Працює швидше і, що найголовніше, підтримується розробниками. [9]

Особливості Parboiled2:

- Дотримується принципів опису граматичних парсерів;
- Генерує однопрохідні парсери. Окремий Лексер не потрібно;
- Використовується типобезпечний DSL, що є підмножиною мови Scala;
- Оптимізації виконуються на етапі компіляції.

Це означає, що можна використовувати parsing expression grammar (PEG) і вільно розбирати рекурсивні структури даних, в той час як регулярні вирази не можуть цього за визначенням. Регулярні вирази не можуть розпарсити навіть JSON та найпростіші арифметичні вирази.

Навіть якщо потрібно розібрати лінійну структуру, Parboiled2 (при використанні належних оптимізацій) буде працювати швидше регулярних виразів. Докази наведено в наступному розділі.

На відміну від генераторів парсерів, таких як ANTLR, не потрібно працювати з роздільною генерацією коду і подальшою його компіляцією. Весь код з Parboiled пишеться на Scala, це дає підсвітку синтаксису і перевірку типів з коробки, так само як і відсутність додаткових операцій над файлами граматик, в той час як парсер, згенерований ANTLR, матиме дві фази синтаксичного розбору. Правда, незважаючи на це, ANTLR все одно потужніший, більш документований і стабільніший, і тому може виявитися кращим у багатьох (дуже нетривіальних) випадках. [10]

Scala parser combinators працюють повільно. Дуже повільно. Маттіас проводив порівняння продуктивності парсерів для Jackson і JSON, написаних за допомогою Parboiled, Parboiled2 і Scala Parser Combinators. [11] З результатами можна ознайомитися далі по тексту.

На відміну від Language Workbenches, Parboiled - маленька і проста у використанні бібліотека. Не потрібно завантажувати погано документоване, повільне ПЗ і витрачати час на пошук потрібних меню і кнопок для опису невеликого DSL. З іншого боку, немає готового текстового редактора з підсвічуванням DSL з коробки, замість цього доведеться самостійно написати плагін для Vim, Emacs або вашої IDE, але це не робить Parboiled менш гідною альтернативою для розробки невеликих предметно-орієнтованих мов. [12]

Parboiled успішно зарекомендував себе в багатьох проектах, в тому числі і в Ентерпрайз. [13]

2.2 Порівняння продуктивності обраних інструментів для парсингу повідомлень HL7 на Scala

Parboiled1 відомий своєю повільністю (у всякому разі, по відношенню до парсером, що генерується ANTLR), викликаної тим, що всі дії в матчингу правил виконувалися в Рантайм і компілятор не міг проводити над таким парсером будь-яких

істотних оптимізацій. У Parboiled2 на чільне місце поставили продуктивність і багато речей були перероблені на макросах, завдяки чому компілятор отримав свободу дій при оптимізації, а користувач - довгоочікувану продуктивність. Нижче продемонстровано, яких результатів домоглися розробники. [14]

Parboiled - це узагальнений інструмент для створення парсеров, а як відомо, спеціалізований інструмент завжди виявляється краще узагальненого в рішенні своєї спеціалізованої завдання.

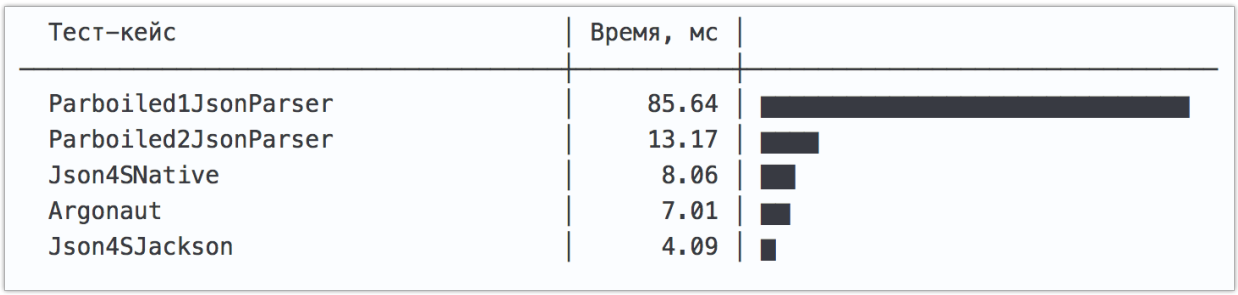

Рисунок 2.1 – порівняння продуктивності Parboiled і стандартних бібліотек для парсингу JSON

Завдяки використанню статичних оптимізацій, Parboiled2 здатний працювати значно швидше регулярних виразів (як мінімум тих, що йдуть в комплекті з бібліотекою класів Java).

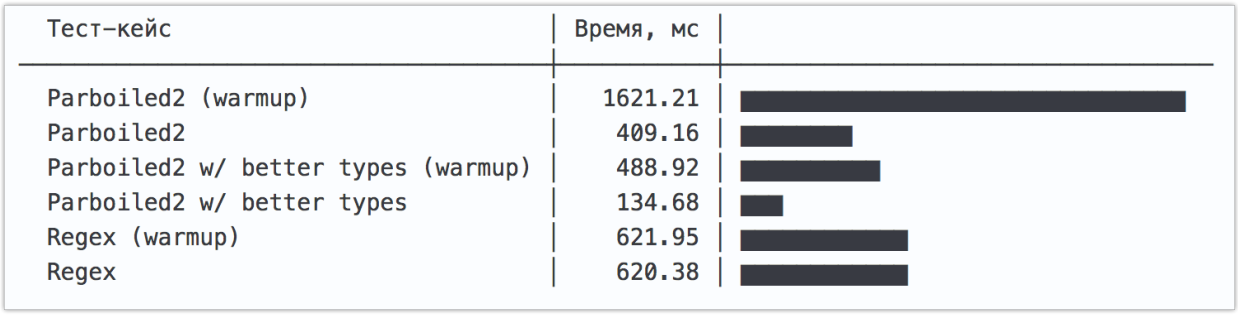

Рисунок 2.2 - Parboiled проти регулярних виразів [15]

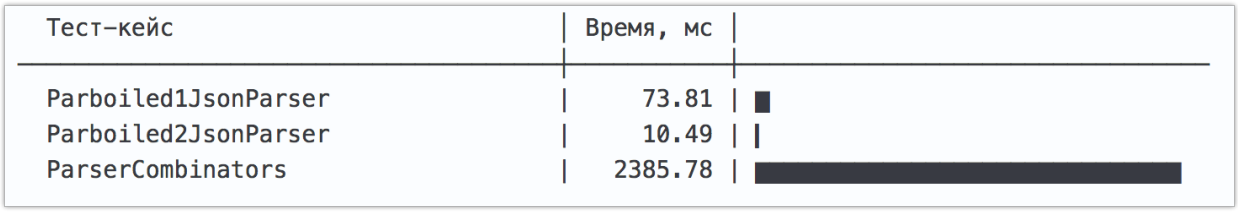

Рисунок 2.3 - Parboiled проти Scala Parser Combinators [16]

Отже, Parboiled2 виявляється найкращим кандидатом на роль парсера у нашому проекті. Він швидкий, гнучкий, має всі можливості що необхідні для написання парсера лінійних виразів, таких як HL7 повідомлення, а також, може допомогти у створенні DSL, для отримання даних.

2.3 Інструменти для побудови DSL для отримання інформації з повідомлення HL7

Єдиною бібліотекою для парсингу HL7 повідомлень на JVM що має широку популярність, є HAPI Terser. Цей синтаксис доволі зручний та випробуваний часом, а також ним користуються, для того щоб мінімізувати зусилля для впровадження нашої бібліотеки, доцільно буде зберегти цей синтаксис якомога ближче, можливо трохи покращивши у деяких місцях. [17]

HAPI Terser обгортає повідомлення, щоб забезпечити доступ до полів за допомогою короткого синтаксису специфікації місцеположення. Наприклад:

```
terser.get("MSH-9-3");
```

Синтаксис специфікації розташування виглядає наступним чином:

```
segment_path_spec "-" field ["(" rep ")"] ["-" component ["-" subcomponent]]
```

Де "rep", "поле", "компонент" та "підкомпонент" - це цілі числа (що представляють, відповідно, повторення поля (починаючи з 0), а також номер поля, номер компонента та числа субкомпонентів (починаючи з 1). Опущення "rep" еквівалентно заданню 0; пропущення компонента або підкомпонента еквівалентно заданню 1. [18]

Синтаксис segment_path_spec має такий вигляд:

segment_path_spec: ["/"] (group_spec ["(" rep ")"] "/")* segment_spec ["(" rep ")"]

Саме таку DSL і матиме наша бібліотека, необхідно зберегти всі функції HAPI Terser для отримання інформації з повідомлення і, частково (можливі певні зміни через технічну можливість).

Також бібліотека має мати 2 інтерфейса: один для безпосередньої роботи на JVM, а інший для мікросервіса.

Для безпосередньої роботи на Scala, доцільно використати стандратні можливості мови, оголосивши неявні перетворення, для того, щоб забезпечити обчислення ланцюжком та розширити елементи методами колекцій. Це дозволить захистити користувача від помилок типів, дозволить використовувати механізми відладки JVM, а також не буде прив’язувати нас до якоїсь стандартної послідовності дій.

Для інтерфейса мікросервіса потрібно використати той самий Parboiled2, що дозволить передавати команди як строкові значення.

2.4 Висновок

Отже, в даному розділі було розглянуто можливості створення синтаксичних аналізаторів на основі опису граматик на платформі JVM, за допомогою можливостей мови Scala.

Було описано результати тестів порівняння Parboiled2, бібліотеки що дозволяє, за допомогою зручного опису граматики, побудувати синтаксичний аналізатор, зі стандартними бібліотеками Scala Parser Combinators, регулярними виразами, а також попередньою версією. Parboiled2 кращий у всіх випадках, а також, лише на декілька мілісекунд відстає від спеціалізованих засобів для парсингу JSON формату, на відміну від попередньої версії. Отже, для реалізації парсера було обрано Parboiled2 як відносно просту, швидку бібіліотеку, що має усі необхідні функції для парсингу повідомлень типу HL7.

DSL буде побудовано на основі HAPI HL7 Terser мови запитів і буде використаовувати для JVM бібліотеки стандартні методи Scala, що дозволяють забезпечити типову безпеку та розширити стандартний набір функцій за рахунок вбудованих колекцій. А для мікросервіса, буде використано той же Parboiled2, для парсингу мови запитів HAPI HL7 Terser.

3 РЕАЛІЗАЦІЯ ПАРСЕРА ПОВІДОМЛЕНЬ HL7

Реалізація складається з двох частин: синтаксичний аналізатор із повідомлення в модель і отримання доступу до інформації за допомогою Domain Specific Language (DSL).

Спочатку буде описана реалізація граматики засобами бібліотеки Parboiled2, а також ієрархічна модель що відображає структуру HL7 повідомлення за допомогою класів Scala.

Другою частиною цього розділу буде опис DSL, а також пояснення реалізації засобами Scala.

Третя частина презентує порівняння роботи отриманого парсера з бібліотекою HAPI HL7 Terser що має схожі цілі і властивості.

Четверта частина присвячена опису API для мікросервіса парсингу даних.

3.1 Опис граматики HL7

Почнемо опис граматики знизу, оголосимо базові компоненти, на яких і буде формуватись граматика.

А саме:

- число;
- строковий рядок;
- спеціальні символи;
- роздільник полів;
- роздільник компонентів;
- роздільник субкомпонентів;
- символ повторення.

Відповідно, відобразимо це в коді:

```
val specialSymbols = """|^~\&"""

val fieldDelimiter = "|"
```

```
val componentDelimiter = "^"

val subComponentDelimiter = "&"

val repetitionSymbol = "~"

def digits = rule { oneOrMore(CharPredicate.Digit) }

def number = rule { digits }

def regularString = rule { oneOrMore(noneOf(specialSymbols)) }
```

Також, HL7 стандарт визначає символьні послідовності, що, при парсингу, мають бути замінені на інші символи, такі як:

Таблиця 3.1 - перетворення спеціальних послідовностей

Символьна послідовність	Результат
\F\	|
\S\	^
\T\	&
\R\	~
\E\	\
\H\	_*
\N\	*_
\Xxx..\	шістнадцятковий формат числа
\.sp <n>\	перехід на наступний рядок n разів
\.sk <n>\	n пробілів

\.br\	перехід на новий рядок

```
def escapeSequence = rule {

 """  \F\""" ~ push("|") |

  """\S\""" ~ push("^") |

  """\T\""" ~ push("&") |

  """\R\""" ~ push("~") |

  """\E\""" ~ push("\\") |

  """\H\""" ~ push("_*") |

  """\N\""" ~ push("*_") |

  """\X\""" ~ capture(regularString) ~ "\\" ~> ((s) => s"0x$s") |

  """\.sp\""" ~ capture(number) ~ "\\" ~> ((n) => "\n" * n.toInt) |

  """\.sk\""" ~ capture(number) ~ "\\" ~> ((n) => " " * n.toInt) |

  """\.br\""" ~ push("\n") |

  "\\" ~ capture(regularString) ~ "\\" ~> ((s) => s"\\$s\\")

}
```

Далі необхідно описати як буде виглядати інформація в повідомленні. Для цього необхідно замінити спеціальні послідовності на те що вони собою представляють, а також відобразити те, що вони можуть бути будь-де.

```
def dataString = rule {
 capture(regularString) ~
   zeroOrMore(
     escapeSequence ~ optional(capture(regularString)) ~>
       ((f, s) => f + s.getOrElse(""))
   ) ~> ((f, s) => {f + s.mkString("")})
}
```

Для наступних кроків, необхідні моделі.

3.1.1 Субкомпонент

Субкомпонент не має особливих параметрів і складається тільки зі значення, оскільки це найменший елемент повідомлення HL7:

```
case class SubComponent(value: String) {
 override def toString: String = value
}
```

Описується він як простий рядок:

```
def subComponent = rule { dataString ~> SubComponent }
```

3.1.2 Компонент

Компонент складається зі списку субкомпонентів і має, щонайменше, один, також, необхідно зазначити, що компонент може складатись із двох субкомпонентів, що оточують пустий субкомпонент, наприклад:

```
|some info&&another info|
```

Необхідно не втратити нумерацію у цьому випадку.

Отже, код буде виглядати наступним чином:

```
case class Component(subComponents: List[Option[SubComponent]]) {

 override def toString: String = { subComponents.map(_.getOrElse("")).mkString("&") }}

def component = rule {

 optional(subComponent) ~

  zeroOrMore(subComponentDelimiter ~ optional(subComponent)) ~>

  ((f, s) => Component(f :: s.toList))

}
```

3.1.3 Поле

Поле складається з списку компонентів, що також можуть бути відсутніми, але впливати на порядок. Поля мають свою особливість: вони можуть повторюватись. Для цієї ситуації підходить патерн проектування композитор для того, щоб описати цю ієрархію. Також, коли помічено символ повторення поля, за ним обов'язково має слідувати поле.

Отже, відобразимо це наступним чином:

```
trait HL7Field

case class RepeatedFields(fields: List[OneField]) extends HL7Field {

 override def toString: String = { fields.mkString("~") }}

case class OneField(components: List[Option[Component]]) extends HL7Field {

 override def toString: String = {
```

```
    components.map(_.getOrElse("")).mkString("^")
  }
}
def fields = rule {
 zeroOrMore(fieldDelimiter ~ optional(field ~ optional(repetitions) ~>
   ((f, s) => {
     s.asInstanceOf[Option[Seq[OneField]]] match {
       case Some(seq) =>
         RepeatedFields(f :: seq.toList)
       case _ =>
         f
     }
   })))
}

def repetitions = rule { oneOrMore(repetitionSymbol ~ field) }

def field = rule {
 optional(component) ~
```

```
  zeroOrMore(componentDelimiter ~ optional(component)) ~>

  ((f, s) => OneField(f :: s.toList))

}
```

3.1.4 Сегмент

Наступною, важливою частиною повідомлення є **сегмент**. У HL7 стандарті це відображається як кожен рядок. Основною ознакою, яку необхідно виділити, є ключ. Також, цей елемент граматики має особливу поведінку, коли ключ “MSH”, це означає, що потрібно ігнорувати поле, що йде після ключа. Представляється цей елемент наступним чином:

```
case class HL7Segment(key: String, fields: List[Option[HL7Field]]) {

 override def toString: String = {

  if (key == "MSH") {

    val head :: tail = fields

    key + head.getOrElse("") + tail.map(_.getOrElse("")).mkString("|")

  } else {

    key + "|" + fields.map(_.getOrElse("")).mkString("|")

  }

 }

}



def segment = rule {
```

```
key ~ optional(specialSymbols) ~ fields ~> ((k, f) => {
  def generateOneField(s: String): OneField = {
    OneField(List(Some(Component(List(Some(SubComponent(s)))))))
  }
  val mshHeader: List[Option[OneField]] =
    if (k == "MSH") {
      List(Some(generateOneField("|")),Some(generateOneField("""^~\&""")))
    } else {
      Nil
    }
  HL7Segment(k, mshHeader ::: f.asInstanceOf[Seq[Option[HL7Field]]].toList)
 })
}
```

І коренем нашої граматики буде саме повідомлення HL7. Представляється він як повідомлення, а також має простий список сегментів і має додатковий метод для повернення всіх ключів. В момент інстанціювання, по повідомленню будується ієрархічна модель на основі синтаксичного аналізатору описаному раніше.

```
case class HL7Message(message: String) {
 val segments: List[HL7Segment] =
  message
```

```
    .split("[\n\r]").toList
    .map(s => new HL7Parser(s).Root.run())
    .zipWithIndex
    .map {
      case (Success(s), _) => s.asInstanceOf[HL7Segment]
      case (Failure(_), n) => throw FailedToParseLineException(s"Failed to parse line: $n")
    }


 def allSegmentsKeys: List[String] = {
   segments.map(_.key)
 }
}
```

3.2 DSL для отримання інформації з повідомлення HL7

Для основи DSL було обрано мову запитів HAPI HL7 Terser, вона є найпоширенішою в JVM середовищі, тому підходить для забезпечення дружнього середовища для розробників, що вже працюють з цим стандартом.[19] Ще однією перевагою буде те, що DSL буде написана з використанням стандартних засобів Scala. Це убезпечить від випадкових помилок за рахунок перевірки під час компіляції.

3.2.1 Сегмент повідомлення

Повідомлення має мати два методи: отримання першого сегменту і всіх сегментів з заданим ключем.

Для отримання першого сегменту будемо використовувати опціональне значення, адже існує варіант коли сегменту з таким ключем не існує в повідомленні, а опціональне значення дозволяє винести логіку роботи з сегментом назовні.

Отже, додаємо такі методи в клас HL7Message:

```
def /(key: String): Option[HL7Segment] = {
  segments.find(_.key == key)
}

def */(key: String): List[HL7Segment] = {
  segments.filter(_.key == key)
}
```

Отримання сегменту по ключу буде відбуватись наступним чином:

```
val message =

"MSH|^~\\&|hl7Integration|hl7Integratio|||||ADT^A01|||2.3|\r" +

"NK1|1|Wood|Father||999-9\r" +
"NK1|2|Jones^Georgy^^^MSS|MOTHER||999-9999"

message / "NK1" shouldEqual Some("NK1|1|Wood|Father||999-9")
```

(message */ "NK1").map(_.toString) should contain theSameElements List("NK1|1|Wood|Father||999-9", "NK1|2|Jones^Georgy^^^MSS|MOTHER||999-9999")

3.2.2 Решта елементів ієрархії

Отримання решти елементів сегмента відбувається за допомогою індекса, де нумерація елементів починається з одиниці. Також існує оператор «*/» для отримання всіх нащадків вказаного дочірнього елемента. Принцип схожий для всєї ієрархічною моделі і його можна представити прикладом отримання компонента та всіх субкомпонентів із поля:

```
def /(index: Int): Option[Component] = {
  components.lift(index - 1) match {
    case Some(s) => s
    case None => None
  }
}

def */(index: Int): List[Option[SubComponent]] = {
  components.lift(index - 1) match {
    case Some(Some(s)) => s.subComponents
    case _ => Nil
  }
}
```

Окрім незначних відмінностей. А саме:

- для отримання поля із сегмента необхідно обробити ситуацію з повтореннями поля;
- отримання субкомпонента із компонента відбувається тільки за допомогою оператора “/”, адже в субкомпонента немає дочірніх елементів.

Але залишилась одна проблема, а саме: пов’язання цих всіх операцій в один запит. Адже, після запита що повертає опціональне значення, не виходить викликати на ньому запит наступного елемента.

Ця проблема вирішується за допомогою використання патерну Scala “pimp my library”, що дозволяє неявно перетворити опціональний елемент в обов’язковий і додати йому функцій. Робиться це наступним чином:

```
object HL7ImplicitConversions {
 implicit def optionSegmentToSegment(os: Option[HL7Segment]): HL7Segment = {
  os match {
   case Some(h: HL7Segment) => h
   case _ => throw NoSuchSegmentException("No such segment")
  }
 }

 implicit def optionFieldToOneField(f: Option[HL7Field]): OneField = {
  f match {
   case Some(h: OneField) => h
   case _ => throw NoSuchFieldException("No such field")
  }
 }

 implicit def optionComponentToComponent(c: Option[Component]): Component = {
  c match {
   case Some(h: Component) => h
   case _ => throw NoSuchComponentException("No such component")
  }
 }
}
```

Таке перетворення визначається для всіх елементів, що повертаються як опціональне значення і мають дочірні елементи.

Цей набір методів та перетворень формує універсальну DSL для роботи з HL7 повідомленнями та отримання будь-якої інформації однозначно. А також, зберігає можливості стандартної мови запитів HAPI і розширює її можливості за рахунок статичної перевірки типів Scala, а також можливості відкладеного обчислення.

3.3 Порівняння розробленого парсера з HAPI Terser

Для порівняння результатів роботи було створено 4 типа повідомлень:

- “Маленьке повідомлення” (8 сегментів, розмір: 2 кілобайта);
- “Середнє повідомлення” (138 сегментів, розмір: 21 кілобайт);
- “Велике повідомлення” (1682 сегмента, розмір: 225 кілобайт);
- “Дуже велике повідомлення” (8382 сегмента, розмір: 1,1 мегабайт).

Для тестування обрано JMH – інструмент для побудови, роботи та аналізу контрольних тестів що написані на мові JVM. [20]

Тест буде проходити наступним чином: буде зроблено ряд запитів різної вкладеності та структури, що покривають собою всі можливості нашої DSL.

Буде також протестовано як поводять себе парсери коли кількість запитів для одного повідомлення збільшується, для того щоб дослідити динаміку зростання кількості операцій. Це реалізовано за рахунок збільшення кількості ітерацій виклику тестового набору запитів. Кількість ітерацій збільшується у 10 разів кожного разу, а також представлено середнє значення операцій за секунду для всіх ітерацій. Також, кожен тест виконується з початковим “прогрівом”, тобто спершу проводиться декілька ітерацій що ніяк не впливають на результат.

Тестовий набір запитів буде виглядати наступним чином:

```
assert((parsedMessage / "PID" / 2).get.toString == "465 306 5961")
assert((parsedMessage / "PID" / 7).get.toString == "19700101")
assert((parsedMessage / "PID" / 5 / 1).get.toString == "Wood")
assert((parsedMessage / "PID" / 5 / 2).get.toString == "Patrick")
assert((parsedMessage / "MSH" / 9 / 1).get.toString == "ADT")
assert((parsedMessage / "MSH" / 9 / 2).get.toString == "A01")

assert(((parsedMessage / "PID" */ 11) (0) / 1).get.toString == "High Street")
assert(((parsedMessage / "PID" */ 11) (0) / 5).get.toString == "Ox1 4DP")
assert(((parsedMessage / "PID" */ 11) (1) / 1).get.toString == "George St")
assert(((parsedMessage / "PID" */ 11) (1) / 5).get.toString == "Ox1 5AP")
```

Отримані дані:

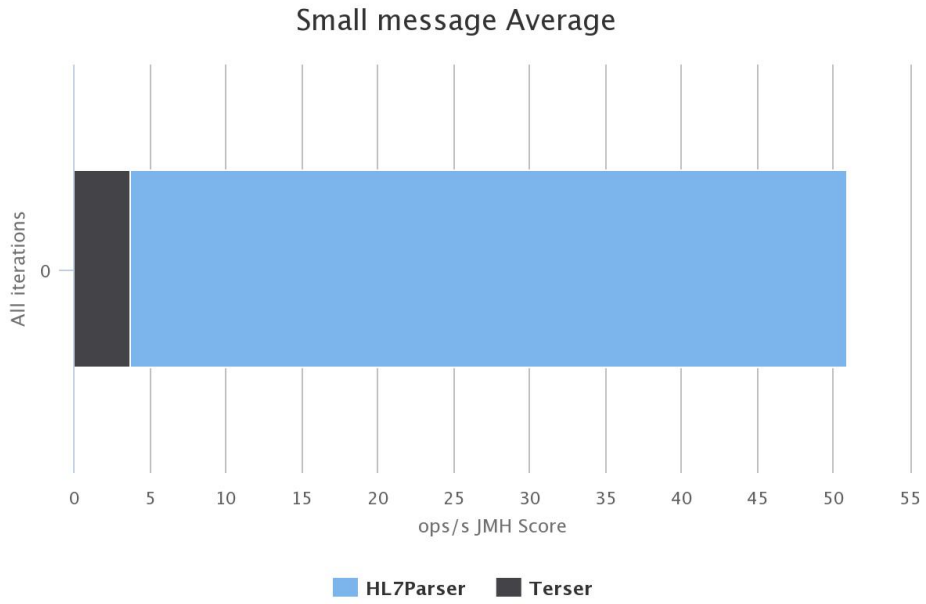

Рисунок 3.1 – середня кількість операцій за секунду для всіх ітерацій у “Маленькому повідомленні”

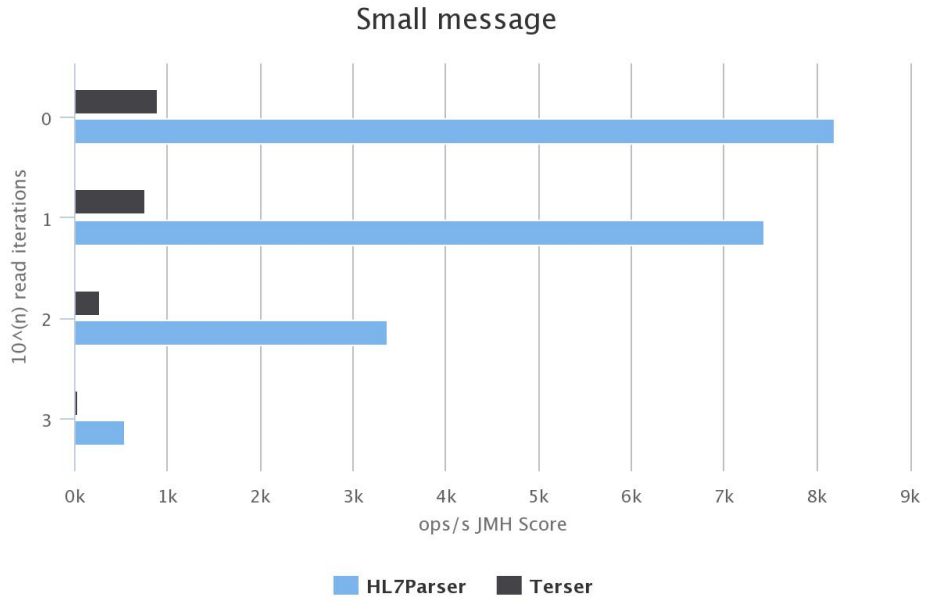

Рисунок 3.2 – кількість операцій за секунду для ітерацій у “Маленькому повідомленні”

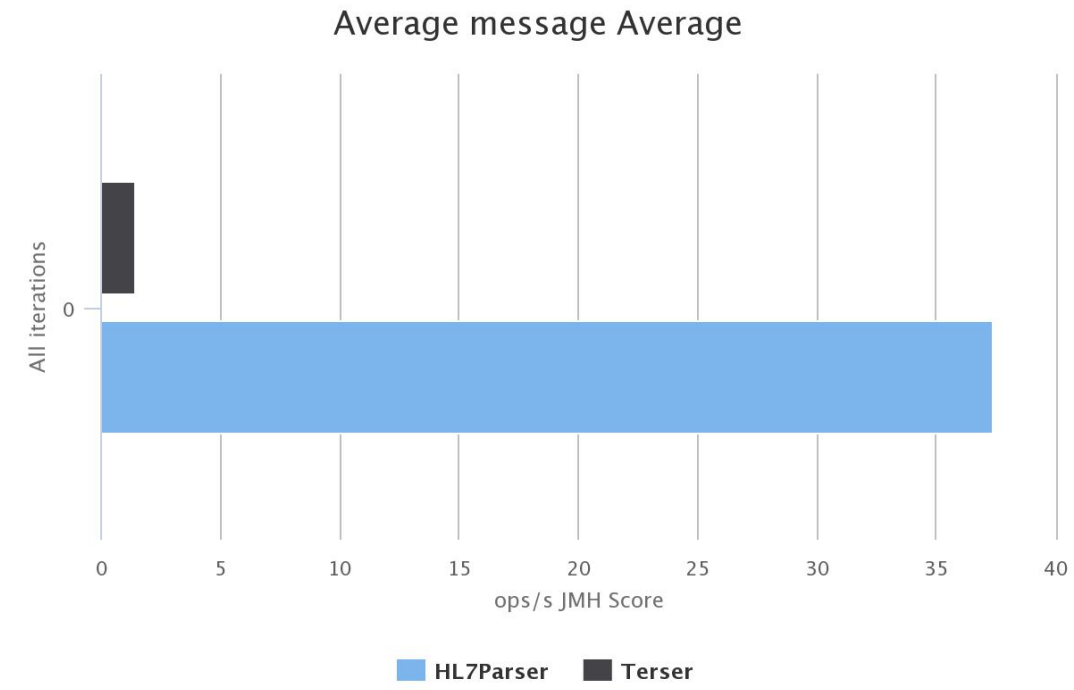

Рисунок 3.3 – середня кількість операцій за секунду для всіх ітерацій у “Середньому повідомленні”

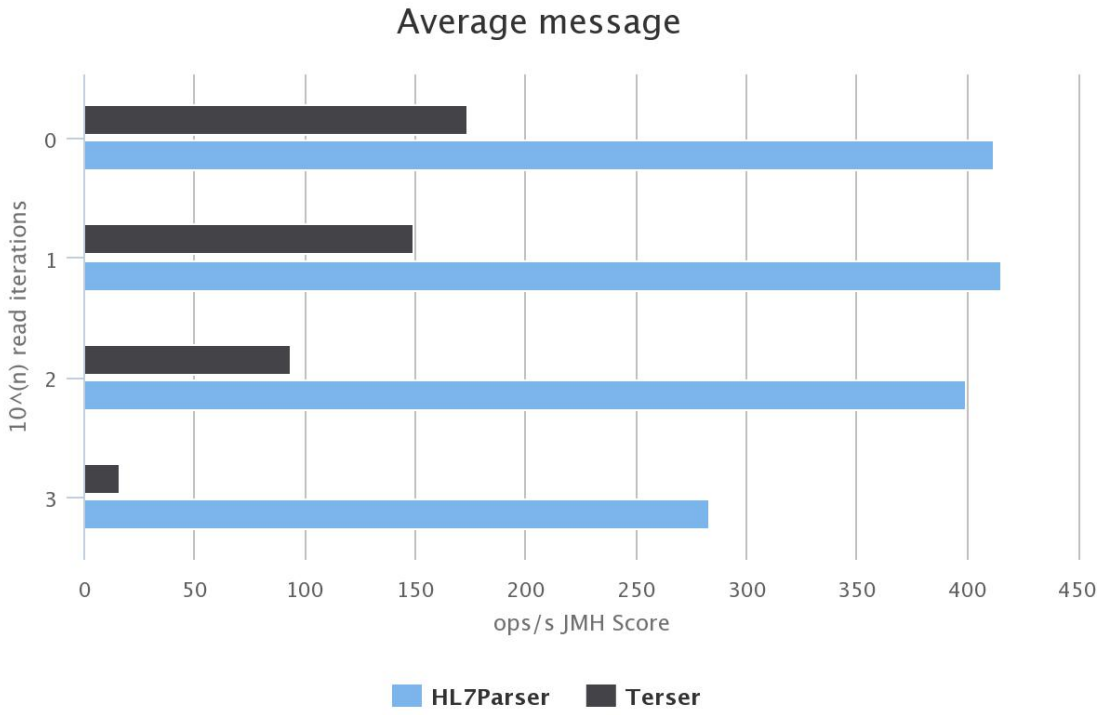

Рисунок 3.4 – кількість операцій за секунду для ітерацій у “Середньому повідомленні”

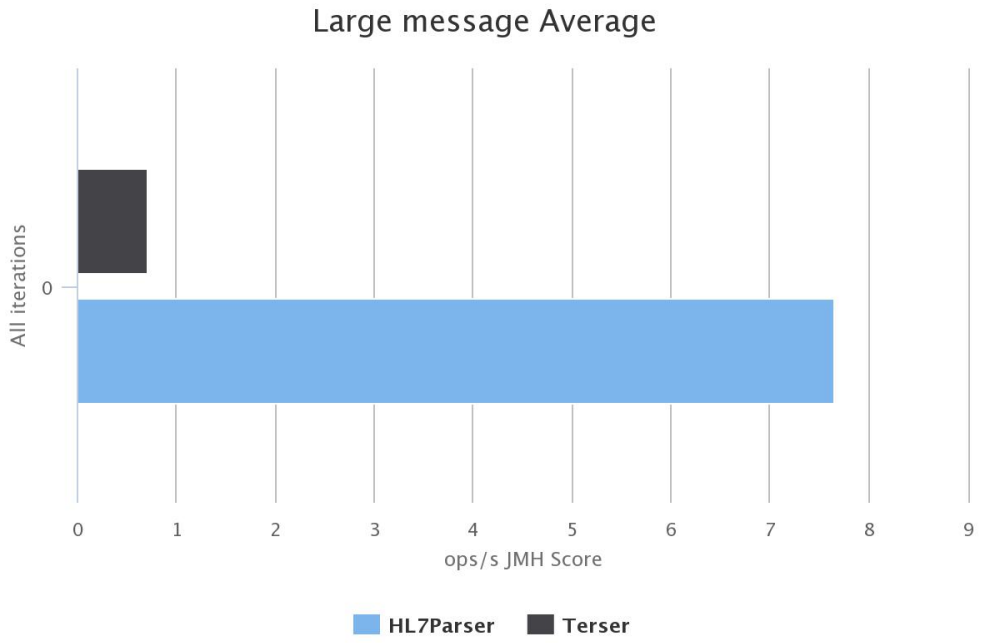

Рисунок 3.5 – середня кількість операцій за секунду для всіх ітерацій у “Великому повідомленні”

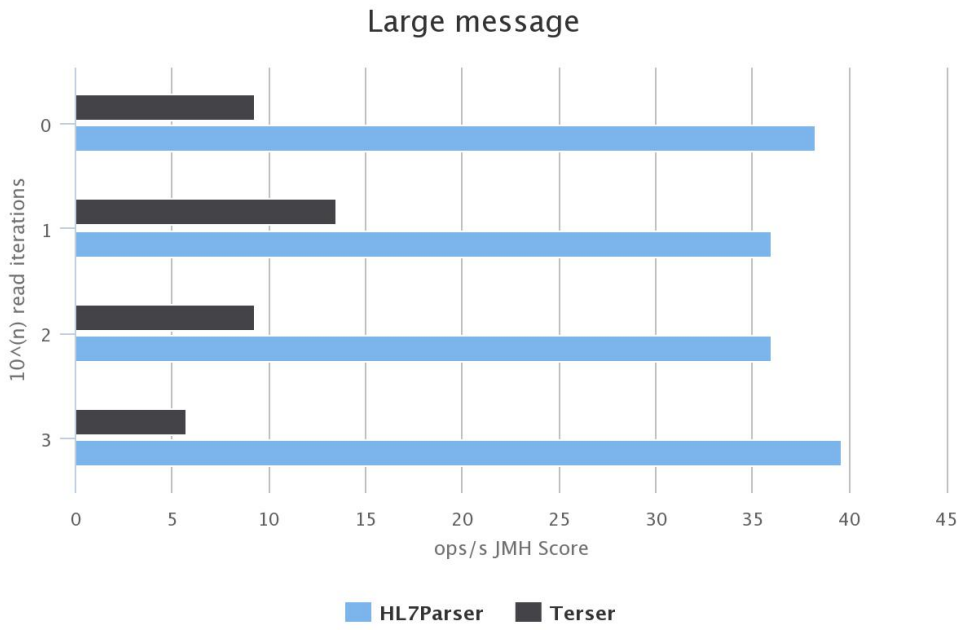

Рисунок 3.6 – кількість операцій за секунду для ітерацій у “Великому повідомленні”

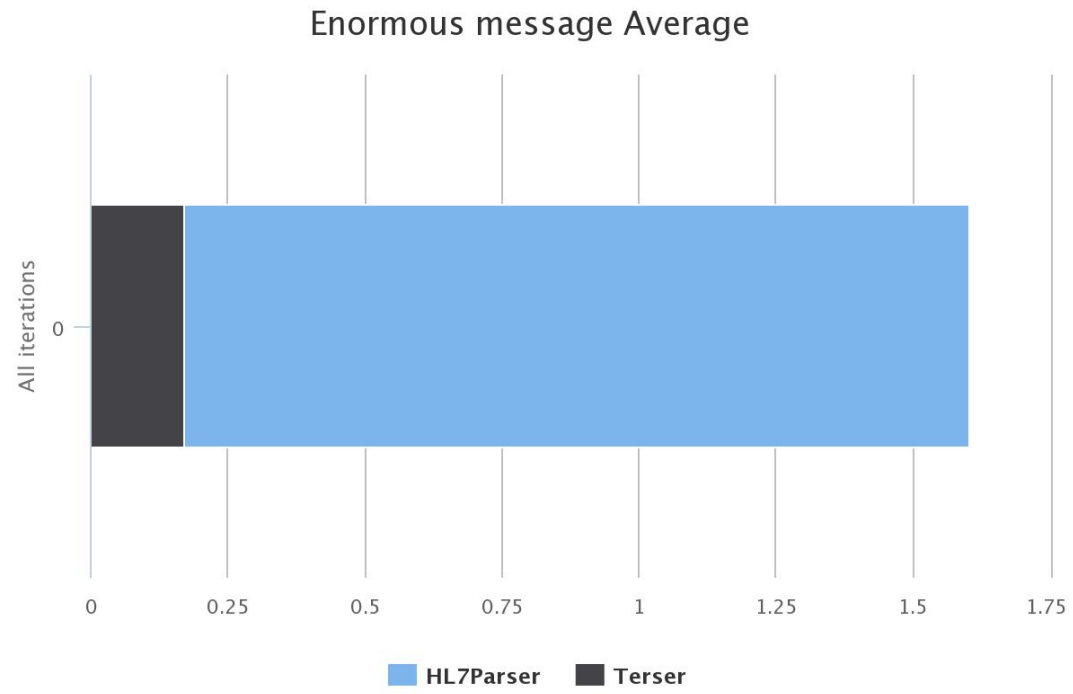

Рисунок 3.7 – середня кількість операцій за секунду для всіх ітерацій у “Дуже великому повідомленні”

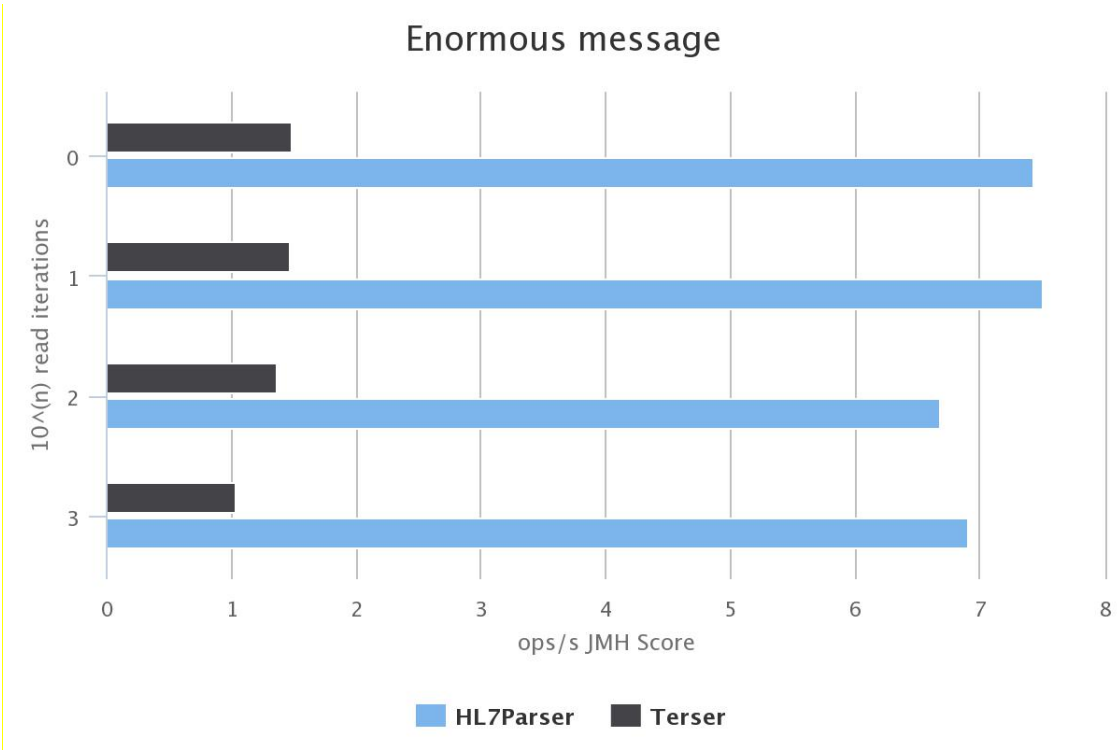

Рисунок 3.8 – кількість операцій за секунду для ітерацій у “Дуже великому повідомленні”

Отже, спираючись на отримані результати, можна сказати що майже у всіх випадках наш парсер разом із DSL працює на порядок швидше ніж найближчий аналог HAPI Terser. Найменший відрив лише на перших ітераціях, але зі збільшенням кількості запитів швидкість різко падає.

3.4 Мікросервіс парсера повідомлень формату HL7

Так як інтерфейс бібліотеки дуже простий, то він реалізується за допомогою Akka HTTP простий Rest сервер з обробкою одного запита, що посилає повідомлення як файл, а також список запитів нашою мовою DSL. У відповідь приходить об’єкт з ключами у формі запитів DSL і значеннями у цих місцях.

Якщо елемента по такому шляху немає – значення буде null.

Парсинг запита найбільш оптимально провести за допомогою регулярних виразів, адже структура буде лінійною і з визначеним порядком.

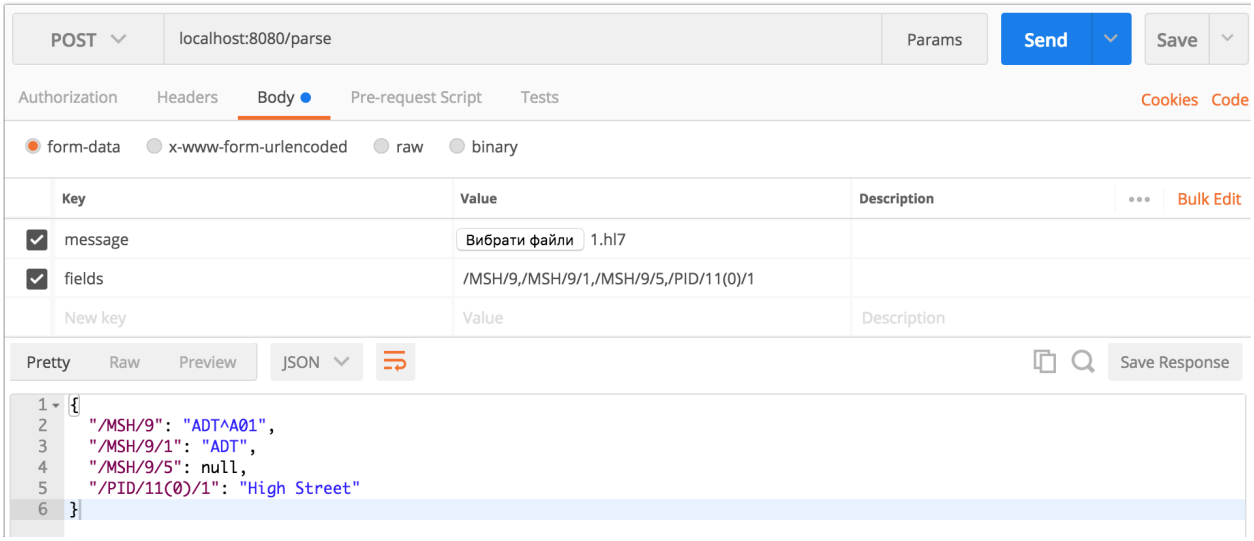

Рисунок 3.9 – Приклад роботи Api

3.5 Висновок

В результаті, була розроблена і реалізована граматика стандарту HL7, а також мова запитів для отримання будь-якої інформації з повідомлення, спираючись на вже існуючу HAPI Terser Query language. За допомогою вбудованих можливостей Scala,

забезпечено гнучкий інтерфейс який дозволяє не тільки покрити всі можливості HAPI Terser, але й додати перевірку типів на момент компіляції, а також інші можливості для роботи з контейнерами.

Наш статичний аналізатор та мова запитів повністю виправдовує своє використання. Вони на порядок швидші за найближчий аналог, легші у використанні, додають перевірки на момент компіляції та додаткові методи для роботи з моделями, дозволяють виконувати всі операції відкладено і, в той же час, повністю покривають функціонал для отримання інформації HAPI Terser'а.

Також було розроблено і протестовано API для мікросервіса, що підвищує кількість можливостей для застосування нашого сервіса.

4 РОЗРОБКА СТАРТАП-ПРОЕКТА «МІКРОСЕРВІС ПАРСИНГУ І АНАЛІЗУ ТЕКСТІВ, ЩО ОТРИМУЮТЬСЯ З ЕЛЕКТРОННОЇ МЕДИЧНОЇ КАРТКИ»

Розділ має на меті проведення маркетингового аналізу стартап проекту “Мікросервіс парсингу і аналізу текстів, які отримуються з електронної медичної картки” задля визначення принципової можливості його ринкового впровадження та можливих напрямів реалізації цього впровадження.

Метою розділу є формування інноваційного мислення, підприємницького духу та формування здатностей щодо оцінювання ринкових перспектив і можливостей комерціалізації основних науково-технічних розробок, сформованих у попередній частині магістерської дисертації у вигляді розроблення концепції стартап-проекту “Мікросервіс парсингу і аналізу текстів, які отримуються з електронної медичної картки” в умовах висококонкурентної ринкової економіки глобалізаційних процесів.

Опис стартап-проекту “Мікросервіс парсингу і аналізу текстів, які отримуються з електронної медичної картки” наведено у Таблиці 4.1.

Таблиця 4.1. Опис ідеї стартап-проекту

Зміст ідеї	*Напрямки застосування*	*Вигоди для користувача*
Створення бібліотеки, що також може бути використана як сервіс. Для швидкого та гнучкого засобу для отримання точкової інформації з повідомлень стандарту HL7	1. Використання як бібліотеки в проекті що використовує JVM.	Використання як бібліотеки безпосередньо дозволяє швидко й гнучко оперувати інформацією, а також оберігає від помилок через неуважність
	2. Використання як мікросервіс через REST API.	REST API надає можливість використовувати систему як сервіс

Отже, проект “Мікросервіс парсингу і аналізу текстів, які отримуються з електронної медичної картки” може бути використано як інструмент для аналізу даних даних з медичної карти, Таблиця 4.2.

Таблиця 4.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

п/п	*Техніко-економічні характеристики ідеї*	*(потенційні) товари/концепції конкурентів*				*W (слабка сторона)*	*N (нейтральна сторона)*	*S (сильна сторона)*
		Мій проект	*Конкурент 1*	*Конкурент2*	*Конкурент 3*			
.	Форма виконання	Веб-сервіс, бібліотека	Бібліотека	Веб-додаток	Веб-додаток			+
.	Кросплатформність	Так	Ні	Так	Так			+
.	Наявність використання безпосередньо в сервісі	Так	Так	Ні	Ні			+
.	Швидкість	Висока	Низька	Низька	Низька			+
.	Горизонтальне масштабування	Так	Ні	Ні	Ні			+
.	Можливість валідації повідомлення на коректність	-	+	+	+	+		

Наш сервіс-бібліотека надає дуже широкі можливості використання у всіх умовах. Але за рахунок того що основний акцент було зроблено на швидкість та зручність, постраждала можливість валідації повідомлення, але в подальшому, її можна буде легко додати. Отож, система є конкурентноспроможною.

4.1 Технологічний аудит ідеї проекту

В межах даного підрозділу необхідно провести аудит технології, за допомогою якої можна реалізувати ідею проекту (технології створення товару) Таблиця 4.3.

Таблиця 4.3 – Технологічна здійсненність ідеї проекту

№ п/п	*Ідея проекту*	*Технології її реалізації*	*Наявність технологій*	*Доступність технологій*
1.	Опис граматики	Parboiled 2	Наявна	Безкоштовна, доступна
2.	Створення DSL	Scala	Наявні	Безкоштовна, доступна
3.	REST API	Akka Http	Наявна	Безкоштовна, доступна

Обрані технології реалізації ідеї проекту: Parboiled 2 через повну безкоштовність фреймворку та швидкість його з поміж конкурентів, наявність досвіду роботи розробників з даною технологією; Scala через простоту використання, наявність досвіду роботи розробників з даною технологією, безкоштовність та можливості для цього завдання; Akka HTTP через безкоштовність і наявність досвіду.

4.2 Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів, Таблиця 4.4.

Таблиця 4.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	*Показники стану ринку (найменування)*	*Характеристика*
1.	Кількість головних гравців, од	5
2.	Загальний обсяг продаж, грн/ум.од	8000 грн./ум.од
3.	Динаміка ринку (якісна оцінка)	Зростає
4.	Наявність обмежень для входу (вказати характер обмежень)	Немає
5.	Специфічні вимоги до стандартизації та сертифікації	Немає
6.	Середня норма рентабельності в галузі (або по ринку), %	R = (3000000 * 100) / (1000000 * 12) = 25%

Отже, було проаналізовано наявність попиту, обсяг, динаміку розвитку ринку. Обмеження для входу на ринок відсутні, динаміка ринку зростає, галузь є рентабельною.

Надалі визначаються потенційні групи клієнтів, їх характеристики, та формується орієнтовний перелік вимог до товару для кожної групи (табл. 4.5).

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проекту

n/n	*Потреба, що формує ринок*	*Цільова аудиторія (цільові сегменти ринку)*	*Відмінності у поведінці різних потенційних цільових груп клієнтів*	*Вимоги споживачів до товару*
.	Необхідне програмне забезпечення(REST API)	Потенційними цільовими групами є дослідницькі центри, університети та компанії, специфіка роботи яких потребує обробки медичних даних	Цільова група займається дослідженнями або має обробляти медичні дані	Рішення повинне бути придатним до інтеграції в інші більш складні системи, бути швидким та гнучким

Згідно проведеної характеристики потенційних клієнтів стартап-проекту випливає, що на ринку є затребуваним програмне забезпечення (REST API та бібліотеки) для доступу медичних даних з карти пацієнта і потенційними цільовими групами є дослідницькі центри, університети та компанії, специфіка роботи яких потребує обробки медичних даних.

Після визначення потенційних груп клієнтів проводиться аналіз ринкового середовища: складаються таблиці факторів, що сприяють ринковому впровадженню проекту, та факторів, що йому перешкоджають (табл. № 4.6-4.7). Фактори в таблиці подавати в порядку зменшення значущості.

Таблиця 4.6 – Фактори загроз

№ п/п	*Фактор*	*Зміст загрози*	*Можлива реакція компанії*
1.	Конкуренція	Вихід на ринок великої компанії	1. Вихід з ринку 2. Запропонувати великій компанії поглинути себе 3. Передбачити додаткові переваги власного ПЗ для того, щоб повідомити про них саме після виходу міжнародної компанії на ринок
2.	Зміна потреб користувачів	Користувачам необхідне програмне забезпечення з іншим функціоналом	1. Передбачити можливість додавання нового функціоналу до створюваного ПЗ
4.	Надходження на ринок альтернативних продуктів	Перехід користувачів нашого товару на інший продукт	Впровадження нового функціоналу, якого немає у конкурентів
3.	Уповільнення росту ринку	Скорочення користувачів продуктів, що тільки виходять на ринок	Інвестиції у впровадження ефективної реклами продукту

Отже, було проаналізовано фактори загроз ринкового впровадження проекту, серед яких: конкуренція, уповільнення росту ринку, зміна потреб користувачів,

надходження на ринок альтернативних продуктів. Було також запропоновано можливі реакції компанії.

Таблиця 4.7 – Фактори можливостей

№ п/п	*Фактор*	*Зміст можливості*	*Можлива реакція компанії*
1.	Стрімкий ріст попиту на інструменти обробки медичних даних	Наявність попиту на інструменти для обробки медичних даних	Змога запропонувати продукт більшій кількості потенційних користувачів
2.	Поява нових різонерів	Надання нового функціоналу для надання результатів роботи нового запитів	Розробка нового функціоналу у вигляді нового HTTP запиту для надання користувачам результатів запитів
3.	Стрімке зростання росту ринку	Компаніям, що тільки виходять на ринок, буде простіше отримати клієнтів	Змога запропонувати продукт більшій кількості потенційних користувачів
4.	Обслуговування додаткових груп споживачів	Поява нових потенційних груп споживачів	Змога розширити продукт для подальшого впровадження у нові галузі
5.	Розширення асортименту можливих послуг	Поява нового функціоналу, що привабить нових користувачів	Розробка нового функціоналу, що є потребою певної групи користувачів

У Таблиці 4.7 наведено фактори можливостей ринкового впровадження проекту, серед яких: стрімкий ріст попиту на інструменти обробки медичних даних, поява нових різонерів, стрімке зростання росту ринку, обслуговування додаткових груп споживачів, розширення асортименту можливих послуг; було також запропоновано можливі реакції компанії.

У наступній таблиці наведено проведення аналізу сильних та слабких сторін стартап-проекту, факторами конкурентоспроможності виступили такі: наявність можливості впровадження як бібліотеки, наявність REST API, швидкість, гнучкість.

Таблиця 4.8 – Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	*Фактор конкурентоспроможності*	*Бали 1-20*	*Рейтинг товарів-конкурентів у порівнянні з нашим підприємством*						
			3	2	1				
1.	Швидкість	20							
3.	Гнучкість	15							
4.	наявність можливості впровадження як бібліотеки	15							
5.	REST API	10							

4.3 Розроблення ринкової стратегії проекту

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів (табл. 9).

Розроблення ринкової стратегії першим кроком передбачає визначення стратегії охоплення ринку: опис цільових груп потенційних споживачів.

Таблиця 4.9 – Вибір цільових груп потенційних споживачів

№ п/п	*Опис профілю цільової групи потенційних клієнтів*	*Готовність споживачів сприйняти продукт*	*Орієнтовний попит в межах цільової групи (сегменту)*	*Інтенсивність конкуренції в сегменті*	*Простота входу у сегмент*
1.	Дослідницькі центри	Спрощення роботи з медичними даними	Великий	Існує 3 конкуренти, які надають схожі, але гірші рішення.	Швидкість, гнучкість
2.	Клініки	Спрощення роботи з медичними даними	Великий		Можливість інтеграції в уже існуючі системи завдяки REST

					API, можливість використання як бібліотеки
Які цільові групи обрано: обираємо клініки, центри контролю якості та дослідницькі центри					

За результатами аналізу потенційних груп споживачів (сегментів) автори ідеї обирають цільові групи, для яких вони пропонуватимуть свій товар, та визначають стратегію охоплення ринку. Для роботи в обраних сегментах ринку необхідно сформувати базову стратегію розвитку. За М. Портером, існують три базові стратегії розвитку, що відрізняються за ступенем охоплення цільового ринку та типом конкурентної переваги, що має бути реалізована на ринку (за витратами або визначними якостями товару).

Отже, проілюструвати базову стратегію розвитку можна у виляді Таблиці 4.10

Таблиця 4.10 – Визначення базової стратегії розвитку

п/п	*Обрана альтернатива розвитку проекту*	*Стратегія охоплення ринку*	*Ключові конкурентоспромо жні позиції відповідно до обраної альтернативи*	*Базова стратегія розвитку*
.	Створення швидкої та гнучкої бібліотеки для отримання точкової інформації з медичної карти	Ринкове позиціонування	Можливість інтеграції в уже існуючі системи завдяки REST API, бібліотеці, швидкість, гнучкість	Диференціація

Було обрано таку альтернативу розвитку проекту: створення веб-сервісу та бібліотеки використовуючи Scala, Parboiled2, Akka HTTP, адже завдяки цим технологіям можна досягнути ключових конкурентноспроможних позицій кінцевого продукту.

Таблиця 4.11 - Визначення базової стратегії конкурентної поведінки

№ п/п	*Чи є проект «першопрохідцем» на ринку?*	*Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?*	*Чи буде компанія копіювати основні характеристики товару конкурента, і які?*	*Стратегія конкурентної поведінки*
1.	Ні	Так	Мова запитів HAPI Terser	Зайняття конкурентної ніші

Отже, було визначено базову стратегію конкурентної поведінки як зайняття конкурентної ніші.

Визначимо стратегію позиціонування у Таблиці 4.12, що полягає у формуванні ринкової позиції (комплексу асоціацій), за яким споживачі мають ідентифікувати торгівельну марку/проект.

Таблиця 4.12 - Визначення стратегії позиціонування

№ п/п	*Вимоги до товару цільової аудиторії*	*Базова стратегі я розвитку*	*Ключові конкурентоспроможні позиції власного стартап- проекту*	*Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)*
1.	Швидка, гнучка з бібліотека з можливістю доступу через REST API	Диференціація	Швидка, гнучка з бібліотека з можливістю доступу через REST API	Швидкість, гнучкість, API

4.4 Розробка маркетингової програми

Першим кроком є формування маркетингової концепції товару, який отримає споживач. Для цього у табл. 4.13 потрібно підсумувати результати попереднього аналізу конкурентоспроможності товару.

Таблиця 4.13 - Визначення ключових переваг концепції потенційного товару

№ п/п	*Потреба*	*Вигода, яку пропонує товар*	*Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)*
1.	Швидкість	Бібліотека на порядок швидша ніж інші	Перевага у швидкості.
2.	Гнучкість	Можливість використовувати бібліотеку для отримання будь-яких даних з HL7 повідомлення	Користувачі отримують гнучкий інструмент для отримання точкової інформації
3.	Наявність можливості впровадження як бібліотеки	Можливість використання звичним способом	Звичний спосіб використання, легкий перехід
4.	REST API	Використання в будь-яких архітектурах	Незалежність платформи

Далі у Таблиці 14 проілюстрована трирівнева маркетингова модель товару: уточнюється ідея продукту та/або послуги, його фізичні складові, особливості процесу його надання.

Таблиця 4.14 - Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Веб-сервіс, що надає доступ до сховища триплетів за допомогою HTTP запитів, дозволяє працювати зі SPARQL-ендпойнтом та надає можливості логічного виведення та хмарного розгортання</td></tr>
<tr><td rowspan="5">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1. Швидкість
2. Гнучкість
3. Наявність можливості впровадження як бібліотеки
4. REST API</td><td>1.Нм
2.Нм
3.Нм
4.Нм</td><td>1.Технологічна
2.Технологічна
3.Технологічна
4.Технологічна</td></tr>
<tr><td colspan="3">Якість: згідно до стандарту ISO 4444 буде проведено тестування</td></tr>
<tr><td colspan="3">Маркування відсутнє</td></tr>
<tr><td colspan="3">Компанія: “IDMC”</td></tr>
<tr><td rowspan="2">III. Товар із підкріплюнням</td><td colspan="3">1-місячна пробна безкоштовна версія</td></tr>
<tr><td colspan="3">Постійна підтримка для користувачів</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: патент</td></tr>
</table>

Було описано три рівні моделі товару, з чого можна зробити висновок, що основні властивості товару у реальному виконанні є нематеріальними та технологічними. Також було надано сутність та складові товару у задумці та товару з підкріпленням.

Після формування маркетингової моделі товару слід особливо відмітити – чим саме проект буде захищено від копіювання. У даному випадку найбільш вірогідним гарантом буде патент.

Наступним кроком є визначення цінових меж, якими необхідно керуватись при встановленні ціни на потенційний товар (остаточне визначення ціни відбувається під

час фінансово-економічного аналізу проекту), яке передбачає аналіз ціни на товари-аналоги або товари субститути, а також аналіз рівня доходів цільової групи споживачів (табл. 4.15). Аналіз проводиться експертним методом.

Таблиця 4.15 - Визначення меж встановлення ціни

№ п/п	*Рівень цін на товари-замінники, грн.*	*Рівень цін на товари-аналоги, грн.*	*Рівень доходів цільової групи споживачів, грн.*	*Верхня та нижня межі встановлення ціни на товар/послугу, грн.*
1.	45000	38000	150000	35000-40000

Наступним кроком є визначення оптимальної системи збуту, в межах якого приймається рішення (табл. 4.16).

Таблиця 4.16 - Формування системи збуту

№ п/п	*Специфіка закупівельної поведінки цільових клієнтів*	*Функції збуту, які має виконувати постачальник товару*	*Глибина каналу збуту*	*Оптимальна система збуту*
1.	Придбання підписки та оплата щомісячних внесків для продовження ліцензії	Продаж	0(напряму), 1(через одного посередника)	Власна та через посередників

Отже, система приноситиме прибуток завдяки щомісячним внескам для подовження ліцензії та придбанням підписок, продаж буде виконуватись напряму або через одного посередника.

Останньою складовою маркетингової програми є розроблення концепції маркетингових комунікацій, що спирається на попередньо обрану основу для позиціонування, визначену специфіку поведінки клієнтів (табл. 4.17).

Таблиця 4.17 - Концепція маркетингових комунікацій

п / п	*Специфіка поведінки цільових клієнтів*	*Канали комунікацій, якими користуютьс я цільові клієнти*	*Клю чові позиції, обрані для позиціону вання*	*Завданн я рекламного повідомлення*	*Концепція рекламного звернення*
.	Придбання ліцензії на користування в мережі Інтернет, щомісячне її продовження, користування сервісом у хмарі або ж на власних серверах.	Інтернет	Інте грація, хмарне розгортан ня, SPARQL-ендпойнт	Показат и переваги сервісу, у тому числі і перед конкурентами	Демо-ролик із використання, рекламні оголошення на популярних сайтах.

Отже, в Таблиці 4.17 наведено концепцію маркетингових комунікацій, було визначено, що придбання ліцензії на користування буде здійснюватись в мережі Інтернет, необхідним буде щомісячне її продовження, користування сервісом можливе у хмарі або ж на власних серверах.

4.5 Висновки

Згідно до проведених досліджень існує можливість ринкової комерціалізації проекту. Також, варто відмітити, що існують перспективи впровадження з огляду на потенційні групи клієнтів, бар‘єри входження не є високими, а проект має дві значні переваги перед конкурентами. Для успішного виходу на ринок у продукту повинні бути наступні характеристики:

- наявність універсального API;
- можливість використання як бібіліотеки;
- швидкість;
- гнучкість.

В рамках даного дослідження були розраховані основні фінансово-економічні показники проекту, а також проведений менеджмент потенційних ризиків. Проаналізувавши отримані результати, можна зробити висновок, що подальша імплементація є доцільною.

Було визначено такі сильні сторони: наявність універсального API , можливість використання як бібіліотеки, швидкість, гнучкість.

Можливості для виходу на ринок включають стрімкий ріст попиту на інструменти обробки медичних даних, можливість впровадження нових різонерів, стрімке зростання росту ринку, обслуговування додаткових груп споживачів, розширення асортименту можливих послуг. Наявні такі фактори загроз: конкуренція, зміна потреб користувачів, надходження на ринок альтернативних продуктів, уповільнення росту ринку.

ВИСНОВКИ

У даній роботі було досліджено основні медичні стандарти медичної карти що широко застосовуються в Європі та Америці, а саме – HL7. Цей формат документообігу став стандартом медичного документообороту в 2003 році і дійсний до сьогодні. На разі існує дуже мало інструментів для роботи з цим стандартом, адже стандарт надає гнучкий формат для обміну даними що дає підприємствам, клінікам, лабораторіям можливість трактувати його по різному, що створює проблему при уніфікації обробки даних. Розглянуто граматику стандарту.

Проаналізовано інструменти для створення і роботи з синтаксичним аналізатором. Також було розглянуто можливі шляхи реалізації парсера з даними властивостями. На основі вже існуючої DSL HAPI Terser DSL для отримання інформації, розроблено свою мову що покриває всі можливі ситуації отримання елемента інформації з повідомлення формату HL7. Під час дослідження виявлено, що Scala як мова програмування і Parboiled2 як засіб для опису граматики для створення синтаксичного аналізатора являються оптимальним інструментом розробки, адже Scala дуже гнучка мова широкого профілю, що, за рахунок підтримки функціональної парадигми, існуючими архітектурними рішеннями для створення DSL, а також підтримці JVM, якнайкраще підходить для розробки синтаксичного аналізатора HL7. Розроблено декілька варіантів взаємодії з запропонованою бібліотекою, а саме: безпосередня можливість звернення як до бібліотеки з платформи JVM, а також REST API для користуванням мікросервісом.

Спроектовано та розроблено синтаксичний аналізатор, що відповідає всім вимогам розглянутими вище. Також було реалізовано два інтерфейса взаємодії з бібліотекою, що робить її незалежною і, водночас, зручною. Перший інтерфейс – безпосередньо Scala інтерфейс, що покриває усі можливі варіанти роботи через JVM мову. Другий – REST API, який дозволяє використовувати бібліотеку як незалежний сервіс та впроваджувати в будь-яку мікросервісну архітектуру.

Проведено тестування створеного рішення. А саме, було створено бенчмарк, що дозволив дослідити динаміку роботи нашого HL7 парсера та HAPI Terser на різних

об’ємах даних. Доведено, що рішення, створене в даній роботі, перевершило аналог у середньому у 8 разів по швидкості знаходження випадкової інформації з повідомлення.

5 ПЕРЕЛІК ПОСИЛАНЬ

1. Institute of Medicine (US) Co2mmittee on Data Standards for Patient Safety Patient Safety: Achieving a New Standard for Care [Електронний ресурс] / Institute of Medicine (US) Committee on Data Standards for Patient Safety – Washington (DC): National Academies Press (US) 2004 – Режим доступу: https://www.nap.edu/read/10863/chapter/7#133 – Дата доступу: 10.04.2018
2. Huff S.M. Clinical data exchange standards and vocabularies for messages [Електронний ресурс] / Huff S.M. – Proc AMIA Symp – 1998 – Режим доступу: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2232190/ – Дата доступу: 13.04.2018
3. Осипов П. HL7 — электронный медицинский документооборот [Електронний ресурс] / Осипов П. – 2012 – Режим доступу: https://habr.com/post/139904/ – Дата доступу: 12.04.2018
4. Сторінка специфікацій HL7 Information [Електронний ресурс] / 2008– Режим доступу: https://blog.interfaceware.com/category/hl7-info/ – Дата доступу: 15.04.2018
5. Big Data статистика мов програмування [Електронний ресурс] / 2017 – Режим доступу: https://www.javaworld.com/article/2846418/big-data/big-data-sparks-interest-in-statistical-programming-languages.html – Дата доступу: 15.04.2018
6. Побудова Abstract Static tree за допомогою Parboiled [Електронний ресурс] / 2017 – Режим доступу: https://www.techopedia.com/definition/22431/abstract-syntax-tree-ast – Дата доступу: 15.04.2018
7. Опис Parboiled для Java [Електронний ресурс] / 2017 – Режим доступу: https://www.hascode.com/2014/01/creating-grammar-parsers-in-java-and-scala-with-parboiled/ast – Дата доступу: 15.04.2018
8. Листування з розробником Parboiled2 [Електронний ресурс] / 2017 – Режим доступу: https://groups.google.com/forum/#!topic/scala-language/sXC-f88Adiw – Дата доступу: 15.04.2018
9. Результати порівняння Parboiled2 з Json парсерами [Електронний ресурс] / 2014 – Режим доступу: http://myltsev.name/ScalaDays2014– Дата доступу: 15.04.2018
10. ANTLR документація [Електронний ресурс] / 2013 – Режим доступу: https://github.com/antlr/antlr4/blob/master/doc/index.md – Дата доступу: 15.04.2018

11. Scala Parser combinators [Електронний ресурс] / 2013 – Режим доступу: http://www.lihaoyi.com/post/EasyParsingwithParserCombinators.html – Дата доступу: 15.04.2018

12. Sebastian Erdweg Evaluating and Comparing Language Workbenches / August 25, 2015 – Режим доступу: http://www.voelter.de/data/pub/LWB-ResultsAndBenchmarks.pdf – Дата доступу: 15.04.2018

13. Проекти що використовують Parboiled [Електронний ресурс] / 2016 – Режим доступу: https://github.com/sirthias/parboiled/wiki/Projects-using-parboiled– Дата доступу: 15.04.2018

14. Parsing Simple Grammars in Scala With parboiled2 [Електронний ресурс] / 2017 – Режим доступу: https://www.threatstack.com/blog/parsing-simple-grammars-with-parboiled2– Дата доступу: 15.04.2018

15. Результати порівняння Parboiled2 з RegExp [Електронний ресурс] / 2014 – Режим доступу: https://groups.google.com/forum/#!msg/parboiled-user/XATcJRLTXjA/XSmf3n6gZSwJ – Дата доступу: 15.04.2018

16. Результати порівняння Parboiled2 з Scala Parser Combinators [Електронний ресурс] / 2014 – Режим доступу: https://groups.google.com/forum/#!topic/parboiled-user/bGtdGvllGgU – Дата доступу: 15.04.2018

17. Приклади використання HAPI Terser [Електронний ресурс] / 2017 – Режим доступу: https://sourceforge.net/p/hl7api/mailman/message/29439275/ – Дата доступу: 15.04.2018

18. Документація HAPI HL7 Terser [Електронний ресурс] / 2012– Режим доступу: https://hapifhir.github.io/hapi-hl7v2/base/apidocs/ca/uhn/hl7v2/util/Terser.html – Дата доступу: 15.04.2018

19. TIOBE Index [Електронний ресурс] / 2018 – Режим доступу: https://www.tiobe.com/tiobe-index/ – Дата доступу: 15.04.2018

20. JMH документація [Електронний ресурс] / 2010 – Режим доступу: http://openjdk.java.net/projects/code-tools/jmh/ – Дата доступу: 15.04.2018